

A Guide to C Resources

SOFTWARE TOOLS FOR ADVANCED PROGRAMMERS

Dr. Dobb's Journal

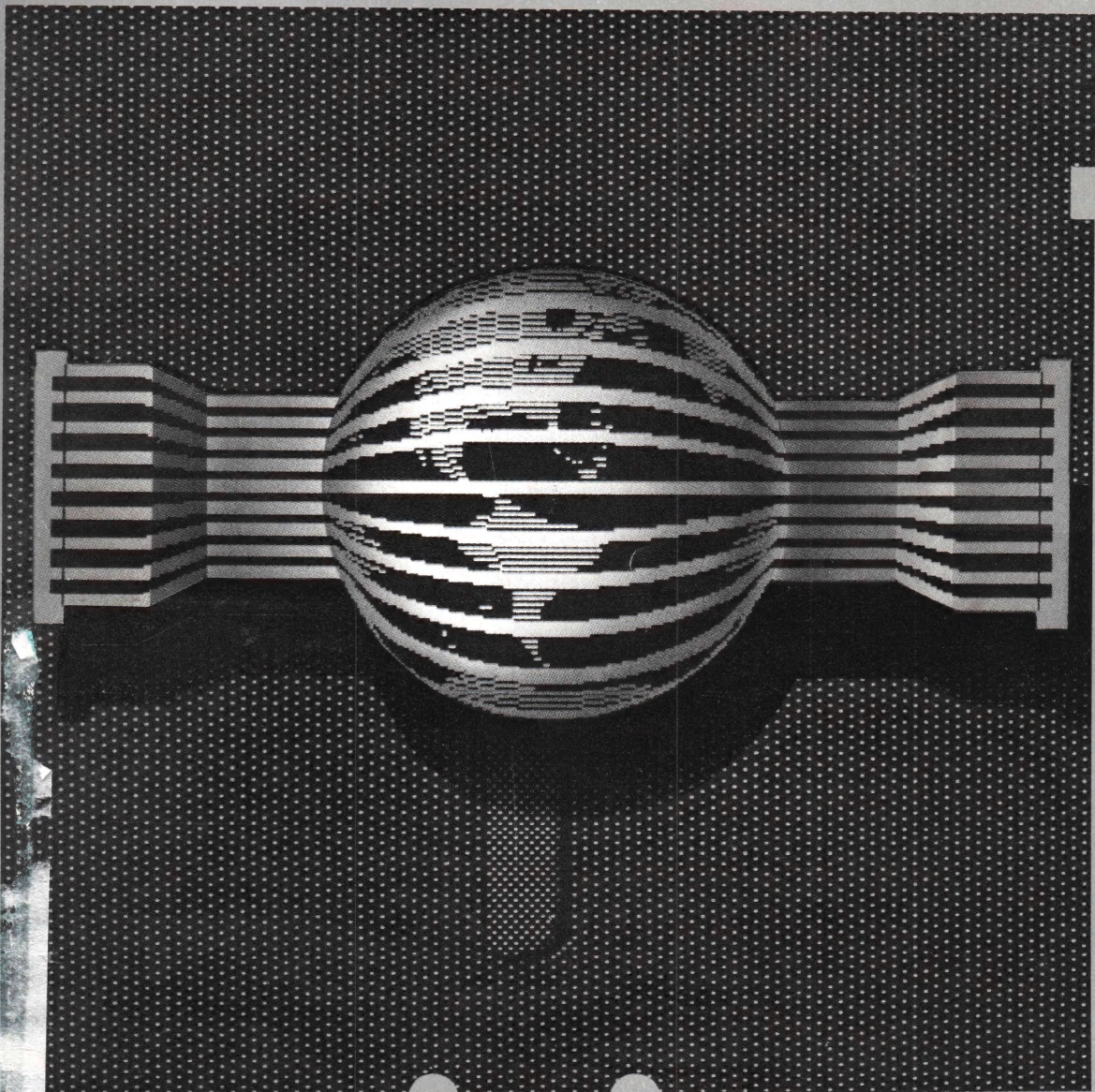
#97 November 1984

\$2.95 (3.50 Canada)

Program Monitor Package

CP/M 2.2 Goes PUBLIC

New Column: Computer Calisthenics



SCREEN SCULPTOR™

**GENERATES CUSTOMIZED
INPUT SCREEN PROGRAMS
IN BASIC AND PASCAL . . .**



THE
SOFTWARE
BOTTLING
COMPANY
OF NEW YORK



BASIC or PASCAL. Easily!

Generate programs in **BASIC** or **PASCAL**. Your choice. Works with * **Turbo PASCAL**, * **IBM PASCAL** or **IBM BASIC** (Interpreter and Compiler). **Easy** to use. Begin productive use in minutes!

POWERFUL SCREENS

Everyone can have professional quality screens to dress up any program. Generate complex, colorful, effective screens in minutes.

FULL FUNCTION SCREEN CREATION

Simply "draw" input screens with word processor style editor.

Advance screen creating features include:

- Draw boxes, lines, etc. in seconds with unique character selection menu.
- Repeat last character in any direction.
- Special color-select screen displays all available colors.
- Paint and Repaint sections of screen at any time.
- Copy and Move sections of screen.
- Insert or Delete characters and lines.
- Input field definition screen gives you total control of character type definitions, edit screen masks, input sequence, variable names, initial values, protected characters, etc.

COMPLETE DATA ENTRY ROUTINES

Generates customized program code that allows professional quality data input using the full PC keyboard (cursor keys, delete, insert, and more). Checks input data for valid entries and displays error messages.

Programs are easily merged with your own programs.

Easy to follow documentation shows how and where you can modify the generated programs.

- Available now with IBM PC, PCjr, PCXT, and all true compatibles.
- Requires 128k RAM, one floppy disk drive, and PC DOS. Works with any 80 column display type.

* Turbo PASCAL is a registered trademark of Boreland International, Ltd. IBM is a registered trademark of IBM corporation.

**TO ORDER SEE YOUR LOCAL DEALER
OR CALL (718) 728-2200**

Only **\$125⁰⁰**

(Includes shipping and handling)
(N.Y.S. Res. Add 8 1/4 % sales tax)

Credit Card Orders call 1-800-824-7888. Operator 268

Alaska and Hawaii call 1-800-824-7919. Operator 268

DEALER INQUIRIES INVITED. SORRY, NO C.O.D.

Produced and Distributed by **The Software Bottling Co. of New York.**
29-14 23 Ave., New York City, N.Y. 11105

ITEM No. 1100



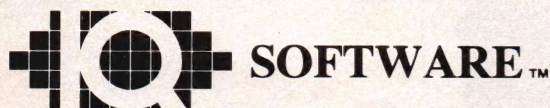
PRESENTS

CP/M[®]

FOR THE

Macintosh[™]

I.Q. SOFTWARE, in one historic move, brings more proven programs to the Macintosh than have existed to date. In the giant world of CP/M based software, **CP/M FOR THE MACINTOSH** delivers the same friendly stand-alone environment to the software developer (professional programmer). If you are wondering why so many developers know and use CP/M it is because CP/M is powerful, easy to learn and easy to use. So easy to use that almost 1 MILLION USERS have CP/M based systems using CP/M based programs and enjoy these same benefits that you will with **CP/M FOR THE MACINTOSH**.



CP/M is a registered trademark of Digital Research, Inc.
Macintosh is a registered trademark of Apple Computer, Inc.

2229 East Loop 820 North
Fort Worth, Texas 76118
(817) 589-2000

Join the network

Develop a program for the new IBM
PC Network.

Promote yourself into the top ranks of network programmers. Write new programs or modify existing ones to run on the local area network that's sure to be a best seller.

Capitalize on a chance to be in from the beginning with integrated business applications, productivity tools and office automation programs.

A high-level interface card makes programming for the IBM PC Network fast and easy. Full data sharing and byte locking capability, multiple servers, two megabits per second data transmission and a complete diagnostic package make the network equally as attractive to the end user.

For complete details about joining the network call 1-800-426-2700. Or write IBM Corporation, Editor, IBM Personal Computer Seminar Proceedings, 5Q9/4629, P.O. Box 1328 Boca Raton, FL 33432.



IBM PC Network Specifications

IBM PC NETWORK ADAPTER

HARDWARE HIGHLIGHTS

MICROPROCESSORS

- 6MHz 80188
- 6MHz 82586

MEMORY

- 32K PROTOCOL ROM
- 16K RAM
- 8K NET BIOS ROM

VIDEO COMPATIBLE RF MODEM

- TRANSMIT 50.75 MHz (CH T14)
- RECEIVE 219 MHz (CH J)
- SUPPORTS 1000 NODES
- MAXIMUM DISTANCE 5KM RADIUS FROM HEADEND
- MULTIPLE SERVICES POSSIBLE

DIAGNOSTICS

- POWER-ON SELF-TEST
- ON-LINE MEDIA MONITORING

OTHER

- SUPPORTS DMA DATA TRANSFERS
- 2-MEGABIT/SECOND DATA RATE
- MID-SPLIT BROADBAND

FIRMWARE HIGHLIGHTS

OPEN ARCHITECTURE

- PEER-TO-PEER NETWORK
- OPERATING SYSTEM INDEPENDENT
- LOCALNET/PC™, PUBLISHED LAYERED PROTOCOL

FUNCTIONS

- BASE FUNCTIONS PROCESSED ON THE ADAPTER, NOT THE PC
- DISTRIBUTED NAME SUPPORT
- REMOTE PROGRAM LOAD
- 32 CONCURRENT TWO-WAY SESSIONS
- HIGH THROUGHPUT RATE AT SESSION LAYER
- CHARACTER SET INDEPENDENT

IBM PC NETWORK CABLING SYSTEM

IBM PC NETWORK TRANSLATOR UNIT

HARDWARE HIGHLIGHTS

- SINGLE RF CHANNEL CONVERSION
- ATTACHMENT OF UP TO 72 PCs WITH IBM CABLING
- ATTACHMENT OF UP TO 256 PCs WITH CUSTOM CABLING
- DATA ONLY
- ALLOWS NODES WITHIN A 1000-FOOT RADIUS

CABLE

TYPE

- STANDARD CATV MEDIA (75 OHM COAX)
- TREE TOPOLOGY
- CATV STANDARD F-CONNECTORS
- PREBALANCED BROADBAND NETWORK

KITS

- BASE EXPANDER (ALLOWS EXPANSION FROM TRANSLATOR)
- SHORT-DISTANCE KIT (1 FOOT ADDITIONAL CABLE)
- MEDIUM-DISTANCE KIT (400 FEET ADDITIONAL CABLE REQUIRED)
- LONG-DISTANCE KIT (800 FEET ADDITIONAL CABLE REQUIRED)
- CABLE AVAILABLE IN 4 LENGTHS: 25 FT., 50 FT., 100 FT., AND 200 FT.

MAXIMUM PCs AND DISTANCES SUPPORT	RADIUS FROM TRANSLATOR	PCS
• TRANSLATOR ONLY	200 FEET	8
• 8 SHORT-DISTANCE KITS	200 FEET	72
• 8 MEDIUM-DISTANCE KITS	600 FEET	72
• 8 LONG-DISTANCE KITS	1,000 FEET	72
• 8-KIT COMBINATION	200 to 1,000 FEET	72

IBM PC NETWORK SOFTWARE

DOS 3.1

EXPANDED SUPPORT FOR NETWORKING

- FILE SHARING
- RECORD LOCKING DOWN TO BYTE LOCKING

PROGRAM INTERFACE TO NETWORK SOFTWARE

- REDIRECTION CONTROL
- INSTALLATION CHECKING
- DIRECT EXECUTION OF NET BIOS FUNCTIONS
- MULTIPLE SERVERS

IBM PC NETWORK PROGRAM

FULL SCREEN INTERFACE AVAILABLE

REDIRECTOR

- ALLOWS USE OF SHARED PRINTERS
- ALLOWS USE OF SHARED DISKS AND DIRECTORIES
- PROVIDES CAPABILITY TO SEND MESSAGES

FILE SERVERS

- SHARED USE OF NAMED DISKS OR SUBDIRECTORIES
- PASSWORD PROTECTION AGAINST UNAUTHORIZED ACCESS
- VARIETY OF ACCESS MODES SUPPORTED (E.G. READ ONLY)
- RECORD LOCKING TO CONTROL MULTIPLE UPDATES

PRINT SERVER

- SHARED USE OF NAMED PRINTERS
- PASSWORD PROTECTION AGAINST UNAUTHORIZED ACCESS
- AUTOMATIC SPOOLING AND QUEUING OF OUTPUT
- QUEUE MANAGEMENT FACILITIES ON SERVER STATION

MESSAGE SERVER

- INTERACTIVE MESSAGE EDITING/TRANSMISSION/RECEPTION
- PRESERVES FOREGROUND APPLICATION CONTEXT
- AUTOMATIC NOTIFICATION OR LOGGING TO DISK/PRINTER

LOCALNET/PC IS A REGISTERED TRADEMARK OF SYTEK, INC.



Dr. Dobb's Journal

Editorial

Editor-in-Chief Michael Swaine
Editor Reynold Wiggins
Managing Editor Randy Sutherland
Contributing Editors Robert Blum,
Dave Cortesi,
Ray Duncan,
Anthony Skjellum,
Michael Wiesenberg
Copy Editors Polly Koch, Cindy Martin
Typesetter Jean Aring
Editorial Intern Mark Johnson

Production

Design/Production
Director Detta Penna
Art Director Shelley Rae Doeden
Production Assistant Alida Hinton
Cover West and Moravec

Advertising

Advertising Director Stephen Friedman
Advertising Sales Walter Andrzejewski,
Shawn Horst
Beth Dudas
Advertising Coordinators Alison Milne,
Lisa Boudreau

Circulation

Circulation and
Promotions Director Beatrice Blatteis
Fulfillment Manager Stephanie Barber
Direct Response
Coordinator Maureen Snee
Promotions Coordinator Jane Sharninghouse
Single Copy Sales
Coordinator Sally Brenton
Single Copy Sales Lorraine McLaughlin
Circulation Assistant Kathleen Boyd

M&T Publishing, Inc.

Chairman of the Board Otmar Weber
Director C.F. von Quadt
President Laird Foshay

Entire contents copyright © 1984 by M&T Publishing, Inc. unless otherwise noted on specific articles. All rights reserved.

Dr. Dobb's Journal (USPS 307690) is published monthly by M&T Publishing, Inc., 2464 Embarcadero Way, Palo Alto, CA 94303, (415) 424-0600. Second class postage paid at Palo Alto and at additional entry points.

Address correction requested. Postmaster: Send Form 3579 to *Dr. Dobb's Journal*, 2464 Embarcadero Way, Palo Alto, CA 94303. **ISSN 0278-6508**

Subscription Rates: \$25 per year within the United States, \$44 for first class to Canada and Mexico, \$62 for airmail to other countries. Payment must be in U.S. Dollars, drawn on a U.S. Bank.

Contributing Subscribers: Christine Bell, W.D. Rausch, DeWitt S. Brown, Burks A. Smith, Robert C. Luckey, Transdata Corp., Mark Ketter, Friden Mailing Equipment, Frank Lawyer, Rodney Black, Kenneth Drexler, Real Paquin, Ed Malin, John Saylor Jr., Ted A. Reuss III, InfoWorld, Stan Veit, Western Material Control, S.P. Kennedy, John Hatch, Richard Jorgensen, John Boak, Bill Spees, R.B. Sutton. **Lifetime Subscribers:** Michael S. Zick, F. Kirk.

Foreign Distributors: ASCII Publishing, Inc. (Japan), Computer Services (Australia), Computer Store (New Zealand), Computercollectief (Nederland), Homecomputer Vertriebs GMBH (West Germany), International Presse (West Germany), La Nacelle Bookstore (France), McGill's News Agency PTY LTD (Australia), Progreso (France).

People's Computer Company

Dr. Dobb's Journal is published by M&T Publishing, Inc. under license from People's Computer Company, 2682 Bishop Dr., Suite 107, San Ramon, CA 94583, a non-profit, educational corporation.

November 1984
Volume 9, Issue 11

CONTENTS

In This Issue

As promised, Michael Wiesenberg starts his column of puzzles for computer solution this month. In addition, we tell you about modifying proprietary software, circumventing the boundaries of user areas in CP/M, determining where your program is spending most of its time, and using a binary tree to maintain a spelling dictionary. There is even a token parsing filter for MSDOS.

Another very useful item is the guide to C programming resources. Until we started working with Terry Ward to get his resource list into production we had little idea how much work such a project can be. As the list grows, maintenance can become as cumbersome as the initial compilation: addresses and product lines change, new companies enter the picture, old companies give up the ghost, new material is published each month . . .

Eventually you just have to decide to stop updating and print the thing. We did succumb, however, to the temptation of last minute additions to the bibliography. While it was too time consuming to do a complete update, we did try to include as much of the latest material from *Dr. Dobb's* as we could. We hope that folks will understand our selectivity in this matter.

Finally, it is almost inevitable that sources will be omitted or come into being after you quit updating. If there are folks who did not get included, drop us a note. If we get enough new information, perhaps we can convince the author to do an addendum.

This Month's Referees

Dr. Dobb's Journal regularly draws on the expertise of a Board of Referees for technical evaluation of material submitted for publication. In addition to remarks to the editors concerning accuracy and relevance of manuscripts, the referees often provide constructive comments for authors regarding clarity or completeness. Their remarks help prevent authors from exposing blindspots or misconceptions in print and help ensure that our readers receive clear and accurate information.

Each month, we print the names of the referees who contributed their insights on material in that particular issue. This space-conserving scheme, unfortunately, does not mention those who work on material that doesn't make it into the magazine. We are in the process of updating our records right now, and will publish another complete list of the board shortly.

The referees who contributed to this month's issue are:

Ian Ashdown, P.E., byHeart Software
Robert Blum, *DDJ* Contributing Editor
David D. Clark, Pennsylvania State University
Michael P. Kelly, Design Software
Douglas W. Rosenberg, Optimal Software
Stan Sieler, Next Generation Systems
Allen Tigert, Krontron Medical Electronics

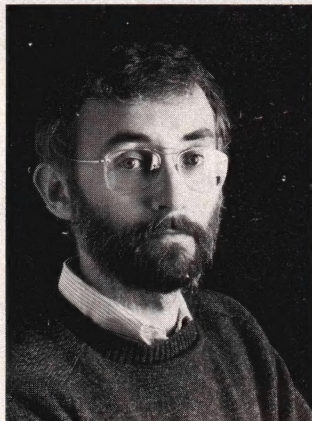
Dr. Dobb's Journal

ARTICLES

- | | |
|--|--|
| <p>Adding Primitive I/O Functions to muLISP
by Michael Carter</p> <p>Program Monitor Package—Using Interrupts to Instrument Applications
by Alan Bomberger</p> <p>CP/M 2.2 Goes PUBLIC
by Bridger Mitchell and Derek McKay</p> <p>A Guide to Resources for the C Programmer
by Terry A. Ward</p> | <p>18 How to add new functions to muLISP, with a look at issues involved in modifying any piece of proprietary software. (Reader Ballot No. 192)</p> <p>26 Optimize intelligently by monitoring where your program is spending most of its time. (Reader Ballot No. 193)</p> <p>48 How to make any file accessible from any user area on a given drive without the duplication necessary to many other schemes. (Reader Ballot No. 194)</p> <p>74 Including a bibliography and lists of program and product sources, this resource guide can help you start tackling the material available. (Reader Ballot No. 195)</p> |
|--|--|
-
- | | |
|--|---|
| <p>RESORT
by Donald G. Krantz</p> | <p>88 This program reads through a document after spelling corrections and uses a binary tree to add words to the user dictionary. (Reader Ballot No. 196)</p> |
|--|---|

DEPARTMENTS

- | | |
|---|---------------------------------|
| <p>Editorial</p> <p>Letters</p> | <p>6</p> <p>8</p> |
|---|---------------------------------|
-
- | | |
|--|---|
| <p>Dr. Dobb's Clinic</p> <p>CP/M Exchange</p> <p>Software Reviews</p> <p>16-Bit Software Toolbox</p> <p>The Software Designer</p> <p>Computer Calisthenics</p> | <p>12 The Sky is Falling, New APL Standard, MacBug, and more (Reader Ballot No. 190)</p> <p>14 Programmer's Tool Chest Improved, CP/M v2.2 PIP Patch (Reader Ballot No. 191)</p> <p>94 Windows for C v2.00, Mychess</p> <p>98 The IBM PC/AT, Counting Cycles, a programming pearl, a token parsing filter, and more (Reader Ballot No. 197)</p> <p>114 A Realizable Fantasy: What programming tools would folks like to see? (Reader Ballot No. 198)</p> <p>116 A new column of puzzles for computer solution (Reader Ballot No. 199)</p> |
|--|---|
-
- | | |
|---|---|
| <p>Book Reviews</p> <p>Of Interest</p> <p>Advertiser Index</p> | <p>119</p> <p>126 (Reader Ballot No. 200)</p> <p>128</p> |
|---|---|



Computer magazines are dropping like dry leaves in a storm. This summer saw the fall of *Personal Software*, *List*, *Computers and Peripherals*, *Color Computer* and the entire Softalk branch: *Softalk*, *Softalk for the IBM Personal Computer*, *St. Mac* and earlier, *St. Game*. Then as the season changed came the announcements that *Microcomputing* and *Microsystems* would cease publication.

I am especially sorry to see those two drop off. Both magazines call up for me an electrically-charged time, days spent playing Little Brick Out on a cassette-based Apple and nights struggling to get a payroll system up on the Board of Health's multiuser Alpha Micro. I grew up in the Midwest, tornado country, and I recall one stormy autumn twenty years ago when my father stopped the car to stand by the side of the road, leaves whirling around his feet, and take pictures of the approaching funnel cloud. The memory of that day came back years later when, between sessions of Little Brick Out, I read early issues of those two magazines and felt another kind of storm brewing: I read in an ionized atmosphere.

Dr. Dobb's advertising sales people see the demise of *Microcomputing* and *Microsystems* as two competitors less. But what do they know.

Microcomputing began with pique and a poke at *Byte* magazine. Born in the late 1970s when the few existing microcomputer magazines were read exclusively by hobbyists, *Microcomputing* had a curious close connection with *Byte*, but to call them sister publications would be seriously to misunderstand the relationship. When *Byte* publishers Wayne and Virginia Green parted company professionally, Virginia took *Byte* and Wayne took it hard. He immediately threw himself into the creation of another computer magazine, which he called *Kilobyte*. What he hoped to do to *Byte* was perhaps a little too obvious in the title; he met with legal objections to its use, and compromised on *Kilobaud*. But however pique-provoked the magazine may have been, its genius was Wayne Green's passion for gadgets, and it spoke to its readers with a freshness and relevance and passion that perhaps only hobbyist magazines can attain. Over the years, *Kilobaud* evolved by way of *Kilobaud Microcomputing* to *Microcomputing*, spinning off a little family group of computer magazines along the way. In 1983, Wayne Green's magazine family was bought by CW Communications, Inc., a huge publisher of computer magazines, and this fall CW decided to retire *Microcomputing*.

Microsystems was originally *S-100 Microsystems*, and was the vehicle through which Sol Libes spread his knowledge about S-100 bus computer systems and products and issues. Libes, even more than Green, was interested in spreading technical information to the community of hackers and tinkerers who evolved with the hardware into professional software developers and system designers. His magazine was always aimed at the advanced user. Eventually it evolved into *Microsystems* and was bought by Ziff-Davis, a major publishing house. Libes was still doing his popular "News and Views" column when, this fall, Ziff-Davis decided that *Microsystems* had no place in its lineup.

It's not that these magazines were perfect or that their publishers were wrong in retiring them. It's just that *Microsystems* and *Microcomputing* were *there*. They lived through, and helped to guide some of us through, a time when the world was changing magically, but the change was still so small you could hold it in your hand; a bright summer of computer hobbydom that is over now. Since I left the Midwest for Silicon Valley I haven't seen dramatic changes of seasons. Except for this one.

Michael Swaine

Michael Swaine

COHERENT™ IS SUPERIOR TO UNIX* AND IT'S AVAILABLE TODAY ON THE IBM PC.

Mark Williams Company hasn't just taken a mini-computer operating system, like UNIX, and ported it to the PC. We wrote COHERENT ourselves. We were able to bring UNIX capability to the PC with the PC in mind, making it the most efficient personal computer work station available at an unbelievable price.

For the first time you get a multi-user, multitasking operating system on your IBM PC. Because COHERENT is UNIX-compatible, UNIX software will run on the PC under COHERENT.

The software system includes a C-compiler and over 100 utilities, all for \$500. Similar environments cost thousands more.

COHERENT on the IBM PC requires a hard disk and 256K memory. It's available on the IBM XT, and Tecmar, Davong and Corvus hard disks.

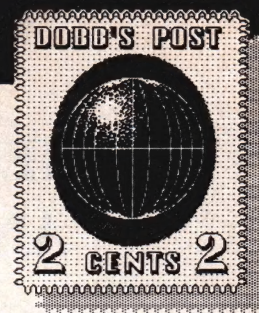
Available now. For additional information, call or write,

Mark Williams Company
1430 West Wrightwood, Chicago, Illinois 60614
312/472-6659



COHERENT is a trade mark of Mark Williams Company.
*UNIX is a trade mark of Bell Laboratories.

Circle no. 76 on reader service card.



On Sixth Generation Computers

Comments on Richard Grigonis' "Sixth Generation Computers" article (DDJ, May 1984) ranged from laudatory to caustic. Michael Doherty's reply to Grigonis the following month drew further comment. Below is a sampling of some of the remarks we received. — Ed.

Grigonis' "Sixth Generation Computers" was stimulating, readable—the best article I've read in a long time!

Robert M. Mason
MRC—130 West Wieuca
Rd, Suite 200
Atlanta, GA 30342

Regarding "Sixth Generation Computers," faster than light travel does not necessarily violate causality. Refer to "Is Faster Than Light Travel Causally Possible?" by Paul Birch, *Journal of the British Interplanetary Society*, v 37, pp. 117–123, 1984.

Peter G. Backes
24016 Kittridge St.
Canoga Park, CA 91307

"Sixth Generation" is too speculative, but interesting. Sixth generation computers will not use methods described. The human brain achieves intelligence without any of these "breakthroughs."

Kjeld Hvatum
P.O. Box 267, MIT Branch
Cambridge, MA 02139

In appreciation of the wonderful sketch of sixth-generation computers by Richard Grigonis (DDJ No. 91) I would like to share with your readers an essay by Miss Bonnie McIntosh:

Since the brain's synaptic signals are much like electronic impulses, a "brain modem" should be developed which could turn my thoughts into

print and graphics on the computer screen. By utilizing this method of communication, thoughts would not be lost or altered by the slow process of verbalizing, drawing, or writing but immediately appear in pure form on the computer screen virtually as fast as they are thought—then edited as desired (a thought processing program?). The brain, through a brain modem, could become a high-speed input device. This process could also be reversed so that individuals could receive information directly from the computer to the brain, a method that could be much quicker than reading or listening to the information. It should also be possible to create or compose music in the same manner, so that the tunes that play "in my head" can be heard by others through brain-to-computer translation.

Peter C. Lincoln, Ph.D.
Bonnie S. McIntosh
Chaminade University of
Honolulu
3240 Waialae Avenue
Honolulu, HI 96816

I enjoyed Michael Doherty's argument very much. We human beings are adaptive for survival, self defense, or greed. Machines, including computers, in the sixth generation still won't possess this inherent adaptive capability.

Dan Ley
84–15 168th Place
Jamaica, NY 11432

Comments to Grigonis' article may be forthcoming, but according to his interpretation of quantum mechanics he should already have access to my comments!

Dr. Jerry W. Lewis
555 E. Boyd Drive, Apt.C
Baton Rouge, LA 70808

This letter is in response to the Grigonis

article "Sixth Generation Computers." It is really a clarification of a point.

As it is that only particles that travel faster than the phase-velocity of light cause the so-called Cherenkov Radiation and that the imaginary tachyons travel faster than the honest-to-god velocity of light, it must be that they do not radiate the aforementioned radiation but must most probably suck in radiation as they travel and as such could be said to emit black-body radiation, which leads us recursively to Max Planck's original theories that were the basis for quantum mechanics; as such, tachyons could be shown to be primitive recursive and therefore could be incorporated in today's computers via the so-called Ackermann function described in the same edition of DDJ.

Thomas Kellar
3104 Hassler St.
Dayton, Ohio 65420

Get rid of Grigonis.

Samuel Hahn
20800 Homestead Rd.
Cupertino, CA 94305

Along the Same Lines . . .

Dear Editor,

Your plaintive praise for the Macintosh user interface (August 1984 editorial) prompts these thoughts. In the future, eyeball-pointing sensors (already mounted on fighter pilot helmets) and voice recognition could provide effortless control of the cursor and word processing functions, eliminating the need for a third hand for the mouse.

The ultimate convenience in user interface may be control by thought, as suggested by the 30-year-old science fiction classic "Forbidden Planet" (MGM, 1956). In this movie, based on Shakespeare's "The Tempest," a central computer with control of immense power generators and matter creation

machines is responsive to the thoughts of the planet's inhabitants. Want breakfast? Just think about food and it appears on the kitchen table. They perished when the computer created the monsters of their subconscious.

Sincerely,
Robert C. Briggs
1337 Rossway Ct.
Los Altos, CA 94022

A Couple of Corrections

Dear Editor:

While keying in Part II of "A New Library for Small-C" (June 1984, *DDJ* No. 92), I encountered a syntax error on page 60. In the ITOD.C function library on line 22, there is a missing semicolon.

The line reads

```
while(sz > 0) str[-sz] = ' '
```

and should read

```
while(sz > 0) str[-sz] = ' ';
```

I've not finished keying the whole library but so far this is the only error I've encountered.

Keep up the good work. I look forward to many months of association with your publication.

Sincerely,
Ken Brayton
9702 Ivanho St.
Spring Valley, CA 92077

[Investigation revealed the most likely culprit to be a runaway knife in the production department. We'll try to keep them under better control. — Ed.]

Dear Editor:

I congratulate you on a most interesting August issue. In reply to the article "What's The Diff" by D. E. Cortesi, please print the enclosed information for your readers. There is a fundamental flaw in the implementation, which I have corrected [see the listing on page 10].

The original program will fail in "pass5" if a block move includes lines that have been affected by "pass4." In fact, whenever a line is duplicated, as happens quite often in structured programming and in Pascal (e.g., begin, end, repeat), the symbol table pointer to any but the last duplicate line is lost.

Still Fixing Bugs The Hard Way?



Ready to take the sting out of debugging? You can with Pfix86™ and Pfix86 Plus™, the most advanced dynamic and symbolic debuggers on the market today for PC DOS and MS-DOS™ programmers.

What other debugger offers you an adjustable multiple-window display so you can view program code and data, breakpoint settings, current machine register and stack contents all at the same time? And, an inline assembler so you can make program corrections directly in assembly language. Plus, powerful breakpoint features that allow you to run a program at full speed until a loop has been performed 100 times, or have the program automatically jump to a temporary patch area.

Or maybe you're tired of searching through endless piles of listings for errors? With Pfix86 Plus you won't have to. You can

locate instruction and data by the symbolic name and using the symbolic address. Handle larger, overlaid programs with ease. And, Pfix86 Plus is designed to work with our Plink86™ linkage editor.

But that's not all. With a single keystroke you can trace an instruction and the action will be immediately reflected in code, data, stack, and register windows. Pressing a different key will elicit a special trace mode that executes call and loop instructions at full speed, as though only a single instruction were being executed.

And you get an easily accessible menu that makes the power of our debuggers instantly available to the new user, but won't inhibit the practiced user.

So, why struggle with bugs? Pfix86 by Phoenix. Pfix86 \$195. Pfix86 Plus \$395.

Call (800) 344-7200, or write.

Phoenix

Phoenix Computer Products Corporation

1416 Providence Highway, Suite 220
Norwood, MA 02062
In Massachusetts (617) 762-5030

Pfix86, Pfix86 Plus and Plink86 are trademarks of Phoenix Software Associates Ltd.
MS-DOS is a trademark of Microsoft Corporation

The required fix is to do the block moves (pass5) before applying Heckel's Rule 2 (pass 4a and 4b).

The author correctly identifies a very inefficient loop at the end of the procedure "resolve." His comment is that an additional pointer must be retained, at a cost of memory. I have changed the logic in this loop so only a single scan of the symbol table is needed.

If memory is at a premium, the size of the symbol table can be reduced to about the same size as the input files! Contrary to intuition, most compares contain only about as many unique lines as the size of one input file. A protective check on the amount of heap

and symbol table space should be added to the procedure "store." I have found that with a TPA of about 55K RAM, the program will compare two 700 line Pascal files.

I think that the author's application of this intriguing algorithm is inappropriate for microcomputers. An ed.com script to change an old file to a new file is not very useful outside the world of mainframe source code updates. Of greater interest is a general utility that displays line differences between two versions of a program, and DIFF can easily be altered to do so.

I do not at all mean to be critical of D. E. Cortesi. Quite the reverse, I

found his article and program fascinating, and my comments reflect the fact that I have spent my time accordingly. Again, thank you.

Steven I. Rothman
President
Copper Creek Systems
P.O. Box 275
One Copper Creek Road
Glenwood, NM 88039

DDJ

Letters Listing (Text begins on page 8)

```
1:
2:  (***** EXCERPT FROM DIFF.PAS *****)
3:
4:  (* PASS5 MUST PRECEDE PASS4!    OTHERWISE LOSE POINTER TO S
5:  Unmatched lines in OA represent deletes; unmatched NA lines are
6:  inserts. Ignoring these, the matched lines should increase
7:  monotonically. When they don't, when a discontinuity appears,
8:  a block-move is present. It is unclear how to record such moves
9:  in a difference file, so here we find them and "unmatch" the
10: moved block, converting it into a delete/insert.
11:
12: There is a figure/ground ambiguity -- any block move can be
13: seen as a move-up of some lines or a move-down of others. We
14: select the smaller of the two groups to be the "moved" group. *)
15:
16: PROCEDURE PASS5;
17: VAR O,N: LINENUM;
18:
19: (* a discontinuity starts at OA[O] and NA[N]. figure out whether
20: fewer lines have been moved up to NA[N] or down from OA[O],
21: and convert the smaller group to inserts and deletes. *)
22:
23: PROCEDURE RESOLVE(VAR O,N: LINENUM);
24: VAR XO,XN,FIRST, LAST: LINENUM;
25:     T: INTEGER;
26:     S: SYMNUM;
27: BEGIN
28:
29: XO:=0; (*measure the block starting at OA[O] *)
30: REPEAT
31:   T:=OA[XO].INDEX + 1;
32:   XO:=XO+1
33: UNTIL (T <> OA[XO].INDEX) OR NOT OA[XO].MATCHED;
34:
35: XN:=N; (*measure the block moved up to NA[N] *)
36: REPEAT
```

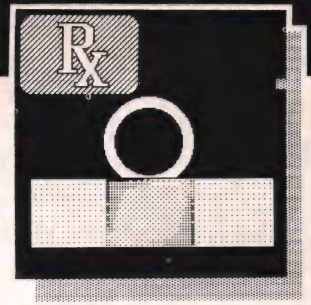


```

37:   T:=NA[XN].INDEX + 1;
38:   XN:=XN+1
39: UNTIL (T <> NA[XN].INDEX) OR NOT NA[XN].MATCHED;
40:
41: (* Unmatch a block of matched lines. We need the symbol table index
42: here, and find it by scanning the table. This expensive operation
43: could be eliminated by retaining the symbol index in OA
44: ABSOLUTELY MUST PRECEDE PASS4, (WHICH CLOBBERS POINTER INTO S)
45: CHANGED FOR AN INEXPENSIVE SINGLE PASS *)
46:
47: IF XO-O < XN-N THEN
48:   BEGIN (*move block down*)
49:     FIRST:=O; (*first oline to convert*)
50:     LAST:=XO-1; (*last oline to convert *)
51:     O:=XO; (*for restart of scan *)
52:   END(*THEN*)
53: ELSE
54:   BEGIN (*move block up*)
55:     FIRST:=NA[N].INDEX; (*first oline *)
56:     LAST:=FIRST+XN-N-1; (*last oline *)
57:     N:=XN; (*restart scan*)
58:   END;(*ELSE*)
59:
60: S:=O;
61: FOR T:=FIRST TO LAST DO BEGIN
62:   WHILE (ST[S].OLINE < FIRST) OR (ST[S].OLINE > LAST) DO S:=S+1;
63:   XO:=ST[S].OLINE;
64:   XN:=OA[XO].INDEX;
65:   WITH OA[XO] DO BEGIN MATCHED:=FALSE; INDEX:=S; END;
66:   WITH NA[XN] DO BEGIN MATCHED:=FALSE; INDEX:=S; END;
67:   S:=S+1;
68: END;(*FOR*)
69: END;(*RESOLVE*)
70:
71: BEGIN(***PASS5***)
72: O:=1; N:=1;
73: REPEAT
74:   WHILE NOT OA[O].MATCHED DO O:=O+1; (* skip deletes *)
75:   WHILE NOT NA[N].MATCHED DO N:=N+1; (* skip inserts *)
76:   IF (N > NEWMAX) OR (O > OLDMAX) THEN EXIT;
77:   IF OA[O].INDEX=N THEN BEGIN O:=O+1; N:=N+1; END
78:   ELSE RESOLVE(O,N); (* discontinuity *)
79: UNTIL FALSE;
80: END;(*PASS5*)
81:

```

End Listing



by D.E. Cortesi, Resident Intern

The Sky Is Falling

Well, maybe not the sky, but most of the world's information processing systems will be falling soon. Or so say Jerome and Marilyn Murray, authors of *Computers in Crisis* (Petrocelli Books, 1984, \$32.95), a copy of which came to this office for review. The Murrays' thesis, as summarized in sensational terms on the book jacket and in its first chapter, is the answer to a computer-illiterate journalist's prayer—an impending disaster of major proportions caused by blind reliance on computers, with overtones of government inertia and commercial callousness. It taps into so many modern archetypes that we predict that the Murrays will soon be appearing on talk shows everywhere.

Since considerably more hot air than illumination will be generated in the ensuing flap, we thought our readers would like a technically literate review of the book.

A FIPS standard determines how data should be stored in computer files. It applies primarily to programs written in COBOL or RPG that run on IBM 370 mainframes. In such programs, decimal numbers usually are stored in what IBM calls packed decimal format: BCD digits with the least-significant four bits reserved for a sign digit. The FIPS standard says dates should occupy four bytes in the form *Oyyymmddss*, where *yy* is the year, *mm* the month, *dd* the day, and *s* the sign. Since these dates are also decimal numbers, they may be compared with each other.

An awful lot of such dates appear in data bases everywhere. The Murrays say that on Monday, January 3, 2000, a lot of programs are going to start doing incorrect things because a date of 0000103 is numerically less, not greater, than one of 0991231. And if nothing is done, they will probably be

proved right. We find it hard to believe that nothing will be done, although this book may serve a useful purpose in focusing attention on the problem sooner than might otherwise happen.

The Murrays' main point is how to fix the problem. Once past the sensational claptrap of the first chapter, the reader finds a sound, interesting tutorial on the history of the calendar and a sensible proposal for a new standard format for dates. The suggested form is *yyyyddss*. This "neo-Julian" date, as packed decimal, fits in the same four bytes as the old version, so you won't have to reconstruct your 20-gigabyte IMS data base.

There you have the meat of the book; we've just saved you \$32.95. The bulk (and we use the word advisedly) of the volume is a set of subroutines for the conversion and manipulation of dates. These are presented in what the authors call pseudo-code. The usual purpose of pseudo-code is to clarify and stress the structure of an algorithm. Unfortunately, the Murrays use a pseudo-code style that is actually *less* structured than typical BASIC (they deny themselves even the use of a for statement).

The subroutines are also presented in 370 assembly language. Fortunately for the publishers, most reviewers won't be able to read IBM ALC. We do know a TRT from a ZAP, and we weren't impressed. The routines aren't reentrant, they use numeric condition codes for branches, they take their arguments in labeled variables not registers, and they commit other coding sins.

The grand catastrophe forecast by this book applies only to programs that store dates in the FIPS format for packed decimal; those that use a different encoding for dates are exempt or at any rate will get no help here. Recommended only for those who work with

software that might be affected—and then only if they can recover the ridiculous cover price from their employers.

APL: A Rigorous Approach

Now, if you want a book with real *body*, let us recommend the *Draft Proposed Standard: Programming Language APL*, available from the ACM Order Department, P.O. Box 64145, Baltimore, MD 21264 for \$25. APL documentation has always been clear and precise, and that tradition has been carried forward even into such anti-linguistic surroundings as an ANSI subcommittee, ordinarily the milieu of the mumble.

That it is written in tight, grammatical English is not the most remarkable thing about this standard. What makes it really unique is that it specifies in full the semantic behavior of an APL implementation. Some standards restrict themselves to syntax; others attempt to specify behavior only for selected parts (as when the Pascal standard struggles to spell out the behavior of files). This standard claims to be "essentially an implementation of APL in a restricted form of English." When we first looked at it, we exclaimed, "Hey, it's a functional spec!" That impression lingers. If you wanted to write your own APL interpreter, you could almost do it by coding the lines of the *Draft Standard* into a programming language. It's that explicit.

Microsoft's MacBug

We asked if newer Microsoft BASICs would still misinterpret the command:

SAVE "PROG,A

Our esteemed editor tried it on his PC and it didn't fail—or at least it didn't write a binary file named PROG.A. But

Richard D. Norling of Washington, DC, tried it on a Mac. "The old bug still lives," he writes, "in the version Microsoft cooked up for the Macintosh. However, this particular bug does not cause further trouble [from the filename]. The Finder, Apple's disk operating system, is capable of handling filenames containing commas."

Norling had another problem, though. He says there is "a killer bug that causes only part of your program to be saved to disk if you touch a key while the save is in progress. No warning, and no way to recover since you aren't likely to discover the problem until you try to load the program at your next session. Microsoft actually issued a new version that fixes this bug in April [but] has left [users] to discover the bug on their own. You are told about the new version only if you call to complain."

We don't see the difficulty. If you keep your hands on the MacMouse where they're supposed to be, you won't be hitting the keyboard during a save anyway, right?

BASIC Haters Unite

Perry Dinsmore of Knoxville, TN, likes controversy and wants to whip some up in this column. We don't mind. All right, Perry, speak your piece.

"Discussions of new computer languages correctly point out the advantages of these languages over older ones such as BASIC. The question remains as to whether there is anything that can be done in a newer language that cannot be done, with a little more effort, in BASIC.

"I would like to challenge anyone to write a program in any language whose end results cannot be duplicated in BASIC. If such a program proves impossible, then a second challenge is to write the non-BASIC program that requires the longest BASIC program to emulate it. For example, if a 12-line program in another language required 120 lines (of comparable length) of BASIC to do the same thing, the expansion factor would be 10. What is the largest possible factor and which other language achieves it?"

If you want to respond to Perry's

challenge, please don't just send a sample of code in Forth, APL, C, or whatever, and say "match that." Life is too short for us, and probably for Perry, to sit around coding up BASIC solutions to other people's programs. You gotta include *both* programs, the short one and the BASIC version you think is equivalent.

Throughputting

We've had 10 or so letters from people who've measured the PIP or COPY throughput of their disk systems. We plan to recap all the numbers in the January Clinic. When you read this, we'll have already written that column,

but there's no reason we wouldn't return to the subject again if more numbers came in. So far we haven't got anything for a Mac nor for any of the appliance machines like Atari or Commodore. Nor for an Apple under any DOS. Comparative throughput figures for a native Apple DOS versus a CP/M softcard would be interesting. We have one extremely impressive figure from a Zenith Z-100 running ZDOS (MSDOS 1.1); it would be nice to have corroboration of it.

DDJ

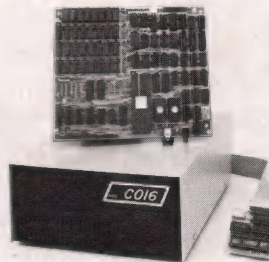
Reader Ballot

Vote for your favorite feature/article.
Circle Reader Service No. 190.

A POWERFUL 68000 DEVELOPMENT ENVIRONMENT FOR YOUR Z80 SYSTEM

CO1668 ATTACHED RESOURCE PROCESSOR

- 68000 Assembler
- C Compiler
- Forth
- Fortran 77



- Pascal
- BASIC-PLUS
- CBASIC
- APL. 68000

6 MHZ 68000 CP/M-68K 768K RAM
4 x 16081 MATH CO-PROCESSORS CPM80 RAM DISK

Develop exciting 68000 applications on your current Z80 based CPM system using powerful mini-frame like 32 bit programming languages. And then, execute them at speeds that will shame many \$100K plus minicomputer systems.

The CO1668 ATTACHED RESOURCE PROCESSOR offers a Z80 CPM system owner a very low cost and logical approach to 68000 development. You have already spent a small fortune on 8 bit diskette drives, terminals, printers, cards cages, power supplies, software, etc. The CO1668 will allow you to enjoy the vastly more powerful 68000 processing environment, while preserving that investment.

CO1668 ATTACHED RESOURCE PROCESSOR SPECIAL FEATURES:

- 68000 running at 6 Mhz
 - 256K to 768K RAM (user partitioned between CPU and RAM Disk usage)
 - Up to four 16081 math co-processors
 - Real time clock, 8 level interrupt controller & proprietary I/O bus
 - Available in tabletop cabinet
 - Delivered w/ sources, logics, & monolithic program development software
 - Easily installed on ANY Z80 CPM system
 - CP/M-68K and DRI's new UNIX V7 compatible C compiler (w/ floating point math) - standard feature
 - Can be used as 768K CPM80 RAM Disk
 - Optional Memory parity
 - No programming or hardware design required for installation
 - Optional 12 month warrantee
- PRICES START AS LOW AS \$899.00 for a CO1668 with 256K RAM, CPM68K, C Compiler, Sources, Prints, 200 page User Manual, Z80 Interface, and 68000 System Development Software.

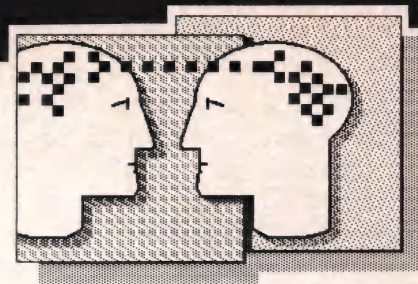
For further information about this revolutionary product or our Intel 8086 Co-Processor, please send \$1 [no checks please] or call:



Hallock Systems Company, Inc.
262 East Main Street
Frankfort, New York 13340
(315) 895-7426

RESELLER AND OEM
INQUIRIES INVITED.

Circle no. 32 on reader service card.



by Robert Blum

I know of nothing more wasteful than setting aside an idea because the difficulty of transforming it into a program seems to outweigh its benefit. The proliferation of data base packages over the past several years proves that I'm not the first and certainly won't be the last to feel this way.

I've tried almost every data base package available for CP/M-80; I prefer to call them application generators. Each has its own unique merits, and all are capable not only of making the transformation from idea to program less time-consuming than a solution based in BASIC or some other high-level programming language, but also of making the job easier to program. For me, however, the benefit of using one of these packages is not great enough to make me switch from assembly language for almost all of my work.

Perhaps I would be singing a different tune if I had not found the right programming tools. Fortunately for me I did, and I couldn't be happier, writing programs encumbered by only my own shortcomings.

Programmer's Tool Chest Improved

SYSLIB, the most complete collection of assembly language subroutines that I am aware of, is now available in its third revision—SYSLIB3. In this release all known bugs in the public domain revision, level 2, have been corrected—there were only a few—and a number of new subroutines have been added. The library has also been restructured to operate completely stand-alone, without any external assistance from other libraries.

Achieving this goal has meant that a few of the subroutines from previous versions are no longer available. They were intended for use in conjunction

with ZCPR, a Unix-like shell written also by Richard Conn, and required support modules from that product to operate properly. In the current release of SYSLIB, all interdependencies between it and ZCPR have been severed. The subroutines contained in SYSLIB are now fully supported within themselves, and any ZCPR-specific subroutines or support modules have been placed into a separate library that is distributed with the ZCPR product.

As I have come to expect, the many new features added to this release of SYSLIB are well thought out and address some deserving areas of the CP/M interface. One of the new features, Switched Output, makes it possible to dynamically switch character output to the console or printer, or simultaneously to both, simply by changing a single memory variable. For example, to display the HL register pair as five decimal characters on the printer requires only the following three instructions:

```
LD    A,80h          ;direct output to
                        ;printer
LD    (SCTLFL),A     ;put into memory
                        ;variable
CALL  SHLDC           ;output HL
                        ;register pair
```

To direct the output to the console or simultaneously to the printer and console is as simple as changing the output direction byte to the proper value and storing it into the memory variable SCTLFL. You can use this same dynamic switching logic for many of the other character output and data format conversion subroutines as well.

The disk I/O routines have been enhanced to include random sector access and buffered single-byte disk I/O, utilizing buffers defined within (and the size chosen by) the application program. You can realize dramatic speed

improvements when data is buffered in memory blocks of at least the same proportion as the physical disk sector size. This is especially true when multiple files are open at the same time. Next month I will go into a more elaborate discussion of how buffered disk I/O works and why the throughput gains are so great, as evidenced by several benchmark tests that I have run.

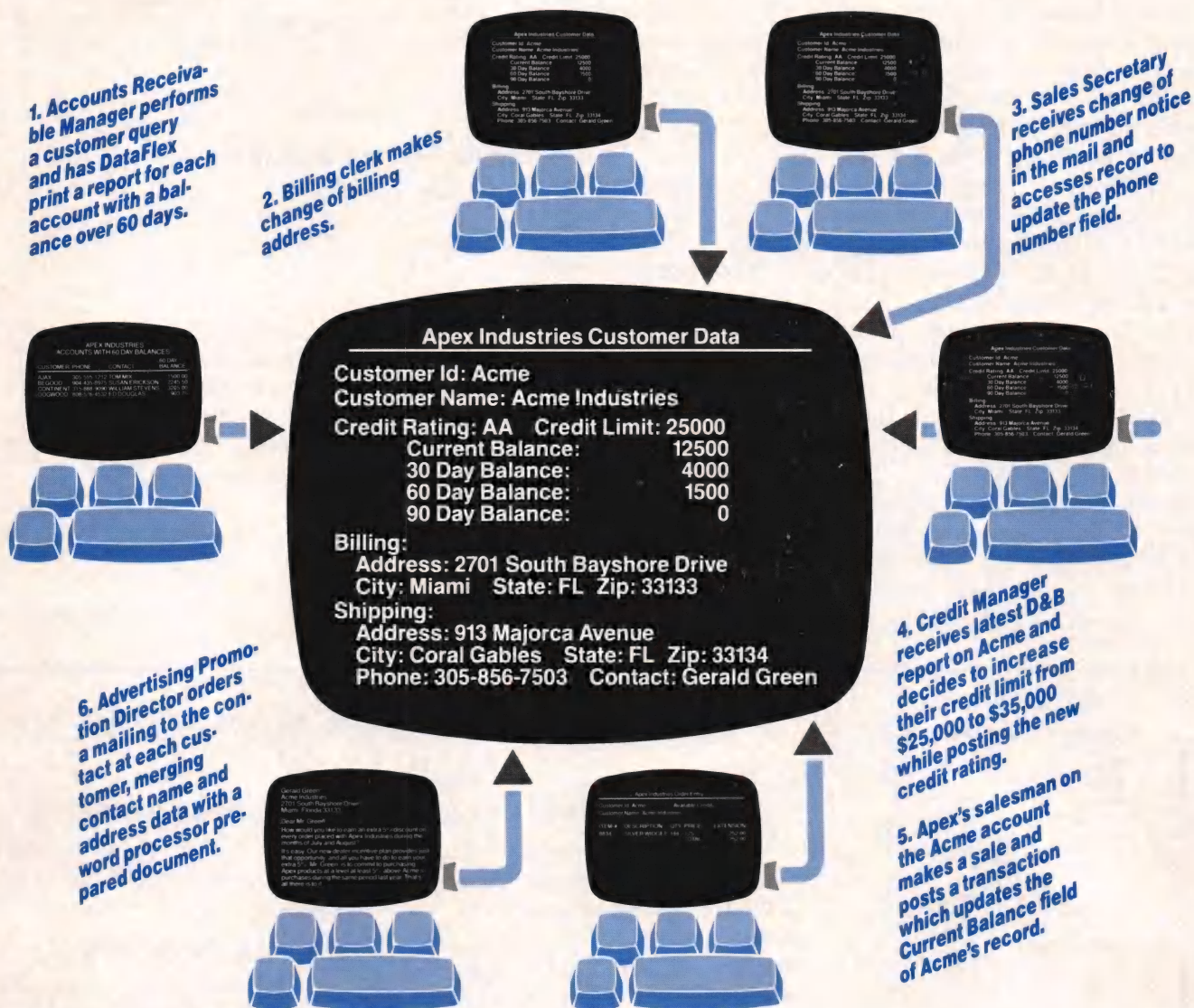
Documentation for SYSLIB now is contained in over 20 on-line help files. Each of the files has been rewritten for this release to explain more completely how to use each of the over 300 subroutines. In prior releases of SYSLIB, the on-line help facility was augmented by a large volume of printed documentation. Although unavailable at this time for SYSLIB3, I understand that efforts are being made to provide printed documentation at a future date for those who desire it.

Release 3 distribution of SYSLIB and ZCPR is being handled through Echelon, Inc., a company headed by Frank Gaudé, the author of DISK7, COMM7, and other notable public domain programs. Included with the purchase of the four-disk set containing SYSLIB is a newsletter subscription dealing with the Echelon family of products. Even though distributed commercially, SYSLIB is priced at only \$29.00, which includes complete source code for each of the subroutines and a license to use the library in any nonprofit way you may desire.

I've been using SYSLIB for over a year now with only favorable results to report. For the experienced programmer, I know of no better way to reduce the coding redundancy inherent to assembly language programming and interfacing to CP/M. For the beginner, I cannot think of a better way to learn. For further information on the products mentioned here, contact Echelon, Inc., 101 First Street, Los Altos, CA 94022.

ALL AT ONCE!

AND NEVER A "LOCKED OUT" USER!



DataFlex is the only application development database which **automatically** gives you true multi-user capabilities. Other systems can lock you out of records or entire files for the full time they are being used by someone else. DataFlex, however, locks only the data being changed, and **only** during the micro-seconds it

takes to actually write it to the file! The updated record is then immediately available. The number of users who can access, and change, records at the same time is limited only by the number of terminals on your system or network. Call or write today for all the details on DataFlex... the true multi-user database.

DATA FLEX

DATA ACCESS CORPORATION

8525 SW 129 Terrace, Miami, FL 33156 (305) 238-0012

Telex 469021 DATA ACCESS CI

See us at Comdex Booth 3349

Compatible with CP/M-80, MSDOS networks, MP/M-86, Novell Sharenet, PC-Net, DMS Hi-net, TurboDOS multi-user, Molecular N-Star, Televideo MmmOST, Action DPC/OS, IBM PC w/Corvus, OMNINET, 3Com EtherSeries and Micromation M/NET.

MSDOS is a trademark of Microsoft. CP/M and MP/M are trademarks of Digital Research.

Circle no. 19 on reader service card.

CP/M V2.2: PIP Patch

Trying to save time by not verifying the output when copying files with PIP will eventually lead to files that cannot be read later on. Rushing seems to be my normal pace, and it began to take its toll in the number of files I was re-creating because I was more intent on getting done than spending a few extra minutes to ensure that my files were secure—that is, until I put into PIP the following patch, which forces the verify (V) option on at all times.

This patch is not approved by DRI; therefore, I suggest that you exercise extreme caution when installing it on your system. The installation steps are straightforward and shouldn't take more than a few minutes, but do be mindful not to overwrite anything of value.

The listing (at right) shows the steps I took to install the verify patch on my system. The first step is to verify that the PIP program being modified is version 1.5; this is the only version of PIP that I know the patch to work with successfully. To verify your version number, first load PIP.COM into memory

with SID or DDT and display memory between 200H and 240H. At 0233H should begin the version number, which must be 1.5. Next change the single byte at 0B34H from 1FH to 37H. Finally save the modified program back onto disk under a new name ready for testing.

To ensure that I had entered my patches correctly, I timed how long it

took to copy the same test file with an unaltered version of PIP using the verify option and again with the newly altered version of PIP. The two resulting times were practically equal, which satisfied me that my modifications were correct.

DDJ

Reader Ballot

Vote for your favorite feature/article.
Circle Reader Service No. 191.

Listing—PIP Verify Patch

```
B>SID PIP.COM
CP/M 3 SID — Version 3.0
NEXT MSZE PC END
1E00 1E00 0100 C8FF
#D200,240
0200 20 20 20 43 4F 50 59 52 49 47 48 54 20 28 43 29 COPYRIGHT (C)
0210 20 31 39 37 39 2C 20 44 49 47 49 54 41 4C 20 52 1979, DIGITAL R
0220 45 53 45 41 52 43 48 2C 20 20 50 49 50 20 56 45 ESEARCH, PIP VE
0230 52 53 20 31 2E 35 03 01 06 01 00 24 24 24 20 20 RS 1.5.....$$$
0240 20
#SB34
0B34 1F 37
0B35 D2 .
#WPIPNEW.COM,100,1E00
003Ah record(s) written.
#GO
```

End Listing

NEW Ver. 2.2
Easier — More Power



WINDOWS FOR C™

FOR THE IBM PC + COMPATIBLES
Lattice C, CI-C86, MWC86
DeSmet C, Microsoft C

C ADVANCED SCREEN MANAGEMENT MADE EASY

ADVANCED FEATURES

- Unlimited windows and text files
- Word wrap, auto scroll
- Horizontal and vertical scroll
- Fast! + No flicker or snow
- No memory in screen buffers
- Complete color control
- Auto memory management
- Save and move window images
- Easy overlay and restore
- Format and print with windows
- Highlighting

WINDOWS++

Much more than a window display system, **Windows for C** is a video display toolkit that simplifies all screen management tasks.

SIMPLIFY • IMPROVE

- Menus
- Data screens
- Form printing
- Help files
- Editors
- Games

ALL DISPLAYS

C SOURCE MODULES FOR
pop-up menus, multiple window
displays, label printer, cursor
control, text mode bar graphs.
plus complete
building block subroutines

DESIGNED FOR PORTABILITY

FULL SOURCE AVAILABLE
NO ROYALTIES

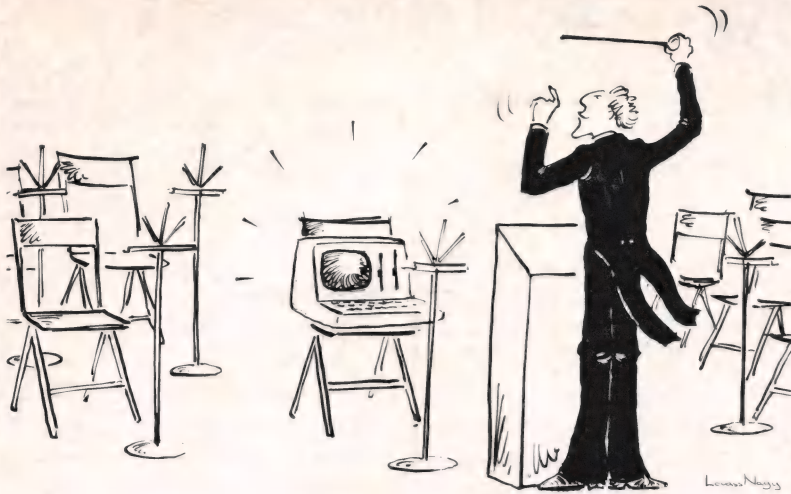
WINDOWS FOR C \$150
(specify compiler & version)
Demo disk and manual \$ 30
(applies toward purchase)
Dealer Inquires welcome

A PROFESSIONAL SOFTWARE TOOL FROM
CREATIVE SOLUTIONS

21 Elm Ave, Box D11, Richford, VT 05476

802-848-7738

Master Card & Visa Accepted
Shipping \$2.50
VT residents add 4% tax



Would you hire an entire band when all you need is one instrument? Of course not.

So why use a whole orchestra of computers when all you need is one to develop software for virtually any type of micro-processor?

The secret? Avocet's family of cross-assemblers. With Avocet cross-assemblers you can develop software for practically every kind of processor — *without having to switch to another development system along the way!*

Cross-Assemblers to Beat the Band!

Development Tools That Work

Avocet cross-assemblers are fast, reliable and user-proven in over 4 years of actual use. Ask NASA, IBM, Xerox or the hundreds of other organizations that use them. Every time you see a new micro-processor-based product, there's a good chance it was developed with Avocet cross-assemblers.

Avocet cross-assemblers are easy to use. They run on almost any personal computer and process assembly language for the most popular microprocessor families.

Your Computer Can Be A Complete Development System

Avocet has the tools you need to enter and assemble your soft-ware and finally cast it in EPROM:

VEDIT Text Editor makes source code entry a snap. Full-screen editing plus a TECO-like command mode for advanced tasks. Easy installation - INSTALL program supports over 40 terminals and personal computers. Customizable keyboard layout. CP/M-80, CP/M-86, MSDOS, PCDOS.....\$150

EPROM Programmers let you program, verify, compare, read, display EPROMS but cost less because they communicate through your personal computer or terminal. No personality modules! On-board intelligence provides menu-based setup for 34 different EPROMS, EEPROMS and MPUs (40-pin devices require socket adaptors). Self-contained unit with internal power supply, RS-232 interface, Textool ZIF socket. Driver software (sold separately) gives you access to all programmer features through your computer, lets you download cross-assembler output files, copy EPROM to disk.

Model 7228 Advanced Programmer — Supports all PROM types listed. Super-fast "adaptive" programming algorithm programs 2764 in 1.1 minutes.

Model 7128 Standard Programmer — Lower-cost version of 7228. Supports all PROM types except "A" versions of 2764 and 27128. Standard programming algorithm programs 2764 in 6.8 minutes.

Avocet Cross-assembler	Target Microprocessor	CP/M-80	CP/M-86 IBM PC, MSDOS**
XASM04 NEW	6804	\$ 250.00	\$ 250.00
XASM05	6805	200.00	250.00
XASM09	6809	200.00	250.00
XASM18	1802/1805	200.00	250.00
XASM48	8048/8041	200.00	250.00
XASM51	8051	200.00	250.00
XASM65	6502/65C02	200.00	250.00
XASM68	6800/01, 6301	200.00	250.00
XASM75	NEC 7500	500.00	500.00
XASM85	8085	250.00	250.00
XASM400	COP400	300.00	300.00
XASMF8	F8/3870	300.00	300.00
XASMZ8	Z8	200.00	250.00
XASMZ80	Z80	250.00	250.00
XMAC682 NEW	68200	595.00	595.00
XMAC68K NEW	68000/68010	595.00	595.00

Model 7956 and 7956-SA Gang Programmers — Similar features to 7228, but program as many as 8 EPROMS at once. 7956-SA stand-alone version copies from a master EPROM. 7956 lab version has all features of stand-alone plus RS-232 interface.

EPROM: 2758, 2716, 2732, 2732A, 2764, 2764A, 27128, 27128A, 27256, 2508, 2516, 2532, 2564, 68764, 68766, 5133, 5143. **CMOS:** 27C16, 27C32, 27C64, MC6716. **EEPROM:** 5213, X2816A, 48016, I2816A, 5213H. **MPU (w/adaptor):** 8748, 8748H, 8749, 8749H, 8741, 8742, 8751, 8755.

7228	Advanced Programmer	\$ 549
7128	Standard Programmer	429
7956	Laboratory Gang Programmer	1099
7956-SA	Stand-Alone Gang Programmer	879
PDV	Driver Software	95
481	8748 Family Socket Adaptor	98
511	8751 Socket Adaptor	174
755	8755 Socket Adaptor	135
CABLE	RS-232 Cable (specify gender)	30

HEXTRAN Universal HEX File Converter — Convert assembler output to other formats for downloading to development systems and target boards. Also useful for examining object file, changing load addresses, extracting parts of files. Converts to and from Intel, Motorola, MOS, RCA, Fairchild, Tektronix, TI, Binary and HEX/ASCII Dump formats. For CP/M, CP/M-86, MSDOS, PCDOS.....\$250

Ask about UNIX.

AVOCET'S SUPERB 68000 CROSS-ASSEMBLER — With exhaustive field testing completed, our 68000 assembler is available for immediate shipment. XMAC68K supports Motorola standard assembly language for the 68000 and 68010. Macros, cross-reference, structured assembly statements, instruction optimization and more. Linker and librarian included. Comprehensive, well-written manual. XMAC682 for MK68200 has similar features.

Call us toll-free for some straight talk about development systems.

1-800-448-8500

(in the U.S. Except Alaska and Hawaii)

VISA and Mastercard accepted. All popular disc formats now available - please specify. Prices do not include shipping and handling - call for exact quotes. OEM INQUIRIES INVITED.

*Trademark of Digital Research **Trademark of Microsoft

AVOCET SYSTEMS INC.™

DEPT. 1184-DDJ
804 SOUTH STATE STREET
DOVER, DELAWARE 19901
302-734-0151 TELEX 467210

Adding Primitive I/O Functions to muLISP

by Michael Carter

One advantage of writing your own software is that you can tailor it to your personal taste. However, writing software is a time-consuming task, and most of us depend upon proprietary packages for most of our software needs. Although the range and quality of software are improving constantly, a proprietary package seldom performs exactly as we want. Usually we conform to the constraints of the package, but occasionally the gap between what we want and what we get is so large that it is worthwhile modifying the package.

This article describes my attempt to overcome a fundamental gap in a proprietary package. Although I deal with a particular example of a particular language, the general principles I discuss will apply to any modification. These principles concern where to locate the modifications, how to pass information to them, and how to recover information from them.

The language that is the subject of this article is LISP. I shall refrain from making a case for the use of this lan-

The Problems

The muLISP implementation of LISP, developed by the Soft Warehouse and distributed by Microsoft, offers more than just a LISP interpreter. It is a LISP program development system complete with its own screen editor. This well conceived and executed package is a joy to use once you get the hang of it. Missing from the package, though, are any facilities for primitive input and output. muLISP has the usual console input and output functions, functions for printer output, and buffered disk I/O, but it provides no way to directly control any of the other I/O ports typically found on a microcomputer. You would need such a facility, for example, to control a robot—a task for which LISP is admirably suited. All that is required is two additional functions: IN[port] and OUT[port,byte].

Fortunately, the authors of muLISP foresaw the need for additional functions and provided a means of linking them into muLISP. They left room for four jump instructions starting at location 103H. Up to three arguments can

“The principles I discuss for modifying proprietary software concern where to locate the modifications, how to pass information to them, and how to recover information from them.”

guage. Suffice it to say that LISP is one of the earliest computer languages and has remained the mainstay of artificial intelligence. LISP or its derivatives will become more common on microcomputers as the applications become more sophisticated.

Michael Carter, Centre for Economic Policy Research, The Australian National University, P.O. Box 4, Canberra, ACT 2600, Australia.

be passed to each function in the HL, DE, and BC register pairs. The value of the function is returned via the HL register pair. (Since muLISP uses address typing to determine a function's type, all machine language routines must begin in low memory; hence the jump table at 103H must be used.)

In attempting to use these facilities, the programmer faces three problems: (1) Where in memory to locate the machine language functions

(2) How to interpret the arguments that are passed

(3) How to return a value to muLISP

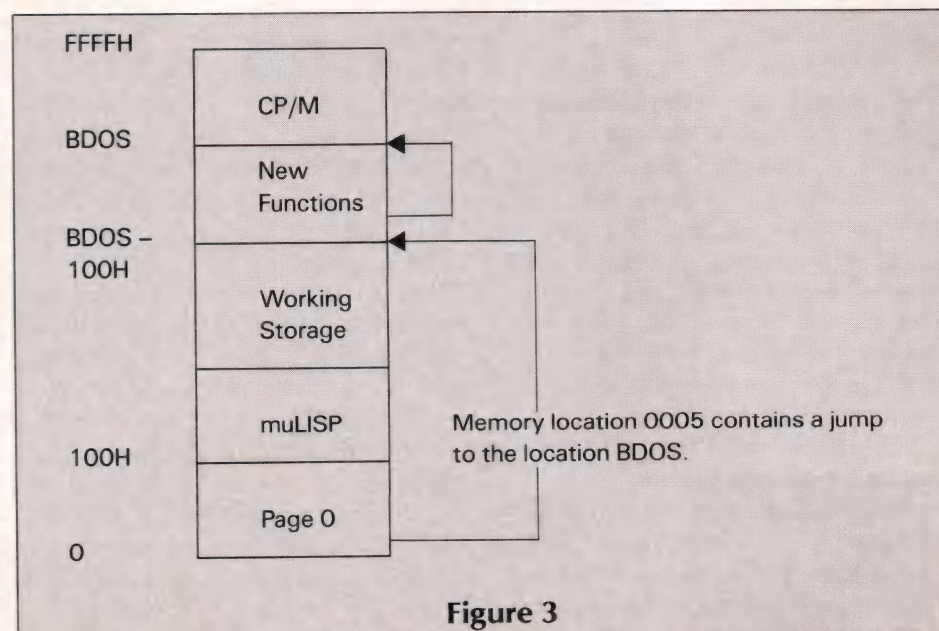
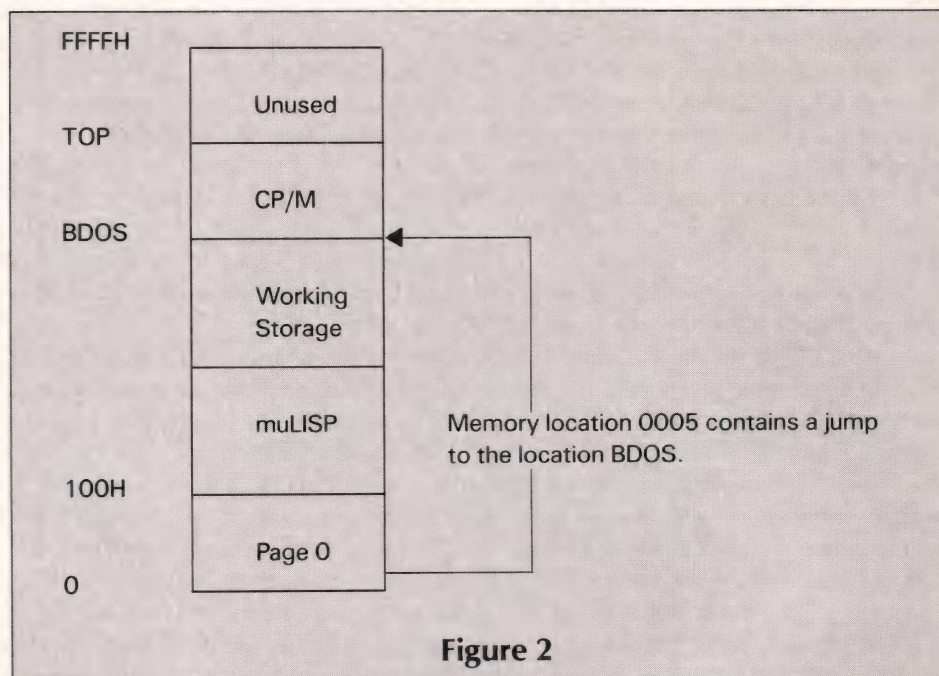
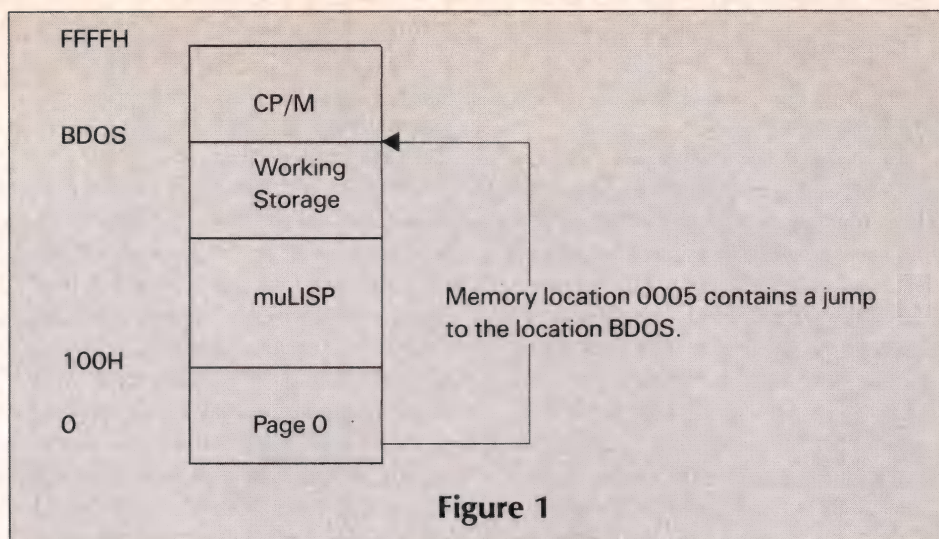
I will elaborate successively on each of these problems and my own solutions. Basically the task is to add two new functions to muLISP (IN and OUT) in such a way that the programmer can regard them as first-class LISP functions.

My discussion of the first problem and its solution is rather elaborate since it has general application and should be of interest to many people who are still becoming familiar with CP/M. However, neither problem (2) nor (3) is specific to LISP or muLISP. Modifying or expanding the facilities of any proprietary program will require some investigation of the means by which information is stored and the suitable adaptation of the package's facilities to new uses. While the particular solution presented here is specific to muLISP, I hope that readers will be able to draw some general principles from my experience that will help them in modifying other packages. Therefore, this article may be considered a general primer on interfacing new functions to existing programs. I used Z80 mnemonics throughout from personal preference, but the only specifically Z80 instruction is the block move LDIR, which you can readily rewrite in 8080 code.

Locating the Functions

The first problem is quite general and one that confronts any programmer attempting to add to the functions of a CP/M program: Where can we find room in the computer for the additional code? The memory map for muLISP running in a standard CP/M system is illustrated in Figure 1 (at right). muLISP uses all the memory between 100H and the BDOS for its own purposes; muLISP itself occupies 10K starting at 100H; and muLISP uses the remainder of the available memory for data storage. This memory usage is typical of CP/M programs.

In some systems CP/M is configured to operate in less than the total available memory, leaving a block of RAM at the top of the memory space (Figure 2, at right). You may use this RAM to hold special-purpose software such as video display or printer drivers, as a printer buffer, or not at all. Whatever



the reason for its existence, any such memory above CP/M could serve to contain the additional muLISP functions.

For obvious reasons most CP/M systems are configured to use all the available memory, and the operating system is located as high in the address space as physically available memory will allow. Remember also that you can readily adjust the size of a CP/M system (line TOP in Figure 2) by running the provided CP/M utility MOVCPM. One solution to our problem would be to create a special version of CP/M for muLISP—a version of the operating system that does not occupy all the available memory, leaving a block of unused RAM to contain the new muLISP functions. The disadvantage of this solution is that we have to cold boot (reset) the system every time we want to run muLISP and every time we finish so that the appropriate version of CP/M is loaded at the right time.

This is not only tedious, it is unnecessary. Rather than creating a special version of CP/M for use with muLISP, we can have muLISP modify the system appropriately whenever it is run. Figure 3 (page 19) illustrates this more convenient and elegant solution. With a little subterfuge we create a “hole” immediately below CP/M to contain the machine language functions so that muLISP thinks it is operating in a slightly smaller system than it really is. Consequently it takes up somewhat less of the available memory, leaving space for the new functions.

All CP/M programs should interface with the operating system by means of a single entry point at memory location 5; that is, programs can utilize the functions provided by CP/M (e.g., write to CONSOLE, open disk file) by a call to location 5 with the desired function encoded in register C. Location 5 in its turn contains a jump instruction to the base of the CP/M BDOS, which also marks the beginning of CP/M in memory (see Figure 1). All space between 100H and this point is available for user programs.

Furthermore, programs (muLISP included) can use the destination of the jump instruction at location 5 to calculate how much memory is available for their own purposes. By modifying the

destination at location 5, we can force muLISP to confine its activities to a smaller region of RAM, thus freeing some memory for the functions IN and OUT (see Figure 3).

Achieving this subterfuge is the first task of the subroutine INIT shown in the listing on page 23. It modifies the destination at location 5 to point to a location 100H lower than the BDOS. When muLISP calculates the size of the available memory, it will use this lower value. In this way we trick muLISP into regarding the 100H bytes of RAM immediately below the BDOS as unavailable, which frees 100H bytes for I/O routines. But how do we persuade muLISP to execute this piece of code?

Like all CP/M programs, muLISP loads and starts execution at 100H. The first instruction is a jump to another location, which we will call START:

```
0100    jmp    START
```

By patching in a jump to INIT at location 100H of muLISP and finishing the subroutine INIT with a jump to START, we convince muLISP to execute the initialization routine before it begins executing its own code.

The address at location 5 serves a dual purpose: it indicates to the user's program the size of the system in which it is running, and it provides a pathway for operating system calls. Therefore, we must provide a further jump to BDOS as the first instruction of this reserved page of memory (see listing). In this way a call to location 5 will find its way correctly to the BDOS, albeit after two jumps. The new functions IN and OUT can immediately follow the jump to BDOS in this reserved page of memory assembled for execution in this location. Locations 103H and 106H are patched to jump to IN and OUT.

This solves the problem of a location for the new functions. But how do we get them there? The simplest way is to tack them on to the end of muLISP following the function INIT and to add a relocater to INIT. The relocater simply moves the code for IN and OUT up to its ultimate destination at BDOS: 100H. These functions are assembled for execution at their ultimate address. This is the purpose of the pseudo-op .PHASE in the listing. This relocation

facility, which is available in the MACRO-80 assembler package, makes this procedure easy. It is not essential, however. The functions can be assembled with ASM at their execution address, loaded with DDT, and then moved down to the end of muLISP for saving on disk.

The patch in the listing is assembled for a particular system size, implied by the address of BDOS. If the available memory changes, the patch must be reassembled to fit the altered system. I chose this implementation for simplicity, since I do not frequently alter my system size. You can overcome this restriction but at the cost of greater complexity. Interested readers might like to consult the articles by Gary Kildall (*DDJ*, February 1978) and John Palmer (*DDJ*, December 1981) on automatic relocation. In this particular example you could easily rewrite IN and OUT in relocatable form since the only absolute reference is the call to GETNUM.

My discussion so far has applied quite generally to CP/M. Now, however, it is necessary to delve more deeply into the structure of muLISP.

Evaluating the Arguments

I said earlier that up to three arguments can be passed to a machine language function in the HL, DE, and BC register pairs. It is important to note that the *value* of the argument is not passed to the machine language function but rather a pointer to its internal representation. To interpret the arguments, you must understand how data is represented internally in muLISP. The following information is taken from Section II of the *muLISP Reference Manual*.

muLISP recognizes three primitive data types: *names*, *numbers*, and *nodes*. Each has a different internal representation. To simplify the machine language functions, I have restricted them to accept and return numbers only. Note that this does not restrict their generality, since any data type can be transmitted as a sequence of numbers. I will confine my attention here to the numbers data type and merely note that the other data types have a similar representation. Those interested in further details should see Section II of the muLISP manual.

A number (in muLISP) consists of

three consecutive pointers:

number	sign	vector
--------	------	--------

A pointer is a 16-bit value (2 bytes) that contains a memory address—a pointer to another location. The first pointer (number) points to itself; that is, it contains its own address. The second pointer or cell (sign) indicates whether the number is positive or negative. The third cell (vector) points to the machine representation of the numerical value of the number.

In contrast to most languages, integers in muLISP are not confined to 16- or 32-bit values. In fact, numbers up to approximately 10^{611} can be represented exactly by allowing the length of the binary representation of a number to be arbitrary and including the length in that representation. The machine representation of the numerical value of a number consists of a single-byte byte counter preceded by the number of bytes required to express the number in binary (including the byte counter). The vector cell is a pointer to this in binary representation—in fact it points to the byte counter. Contrary to the manual, the byte counter follows rather than precedes the bit vector, which itself is stored most significant bit first. The number zero is a special case; it is represented by a single byte, the byte counter, which has the value one.

The argument that is passed to a machine language function in one of the register pairs (assuming a numerical argument) is a pointer to the number cell of the internal representation of the number. For example, suppose that the first argument is the value six. Then the register pair (HL) will point to a muLISP number that has the value six. The internal representation of this is depicted in Figure 4 (page 22).

We are now in a position to understand the subroutine GETNUM in the listing. This subroutine returns the numerical value of the argument that is pointed to by the HL register pair. GETNUM skips over the number and sign cells to the vector cell. The value of this pointer is loaded into the HL register pair, which now points to the byte counter. If the value of the byte counter is one, the value of the number is zero, which GETNUM returns. If

Debugging Bugging You?

Torpedo program crashes and debugging delays with debugging dynamite for the IBM PC ...

UP PERISCOPE!

First, you install the hardware.

The hardware's a special memory board that fits in a PC expansion slot. Its 16K of write-protected memory contains Periscope's resident symbolic debugger. No runaway program, however berserk it may be, can touch this memory!

Then you UP PERISCOPE.

Use Periscope's push-button break-out switch to interrupt a running program ... even when the system's hung! Periscope supports Assembly, BASIC, C and Pascal. In addition to the usual debugging capabilities, some of Periscope's features are:

Stop your system in its tracks at any time.

Use symbol names instead of addresses.

Run a program on one monitor and debug on another.

Monitor your program's execution with Periscope's comprehensive breakpoints.

Debug memory-resident programs.

Put your time to better use.

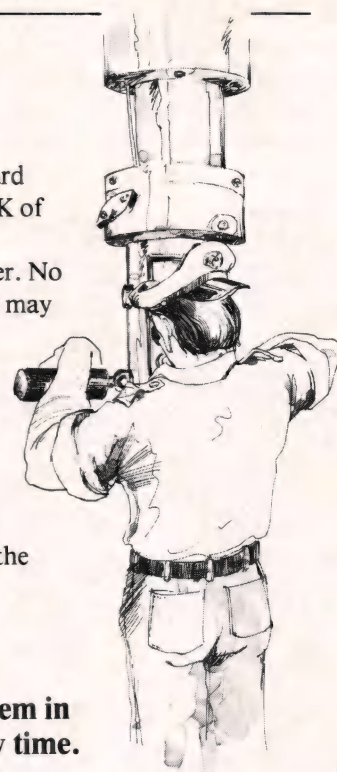
The Periscope system is \$295. It carries a 30-day money-back guarantee and includes the memory board, remote break-out switch, debugger software, 100-page manual, and quick-reference card. The memory board is warranted for one year. A demonstration disk is \$5.00.

System requirements for Periscope are an IBM PC, XT or Compaq, PC-DOS, 64K RAM, 1 disk drive and an 80-column monitor. For MasterCard and Visa orders only, call 800/421-5300 (ext. R96) 24 hours a day. For additional information, call 404/256-3860 from 9 AM to 5 PM Eastern Time.

Get your programs up and running;

UP PERISCOPE!

Data Base Decisions / 14 Bonnie Lane / Atlanta, GA 30328



not, GETNUM returns the value of the immediately preceding byte.

In this particular application all numbers (port numbers and byte values) lie between 0 and 255. You may substantially simplify GETNUM by taking advantage of this knowledge. Thus, it is implicitly assumed that the byte counter will have a value of one or two. If not, GETNUM returns the val-

ue of the number mod 256.

Returning a Value to muLISP

The value of a machine language function is returned via the HL register, but the HL register does not contain the value itself. Rather it contains a pointer to the appropriate data structure. In the case of the input function, this structure will be the address of a bona fide mu-

LISP number; that is, the input function must return a pointer to a sequence of three cells (number, sign, and vector), the last of which contains the address of the binary representation of the desired value. Achieving this turned out to be a nontrivial task, and it is worth exploring briefly two false paths I followed for the insight they provide into the workings of the interpreter.

My first attempt was to build up the appropriate data structure as part of the code of the I/O functions themselves. IN had the form shown in Figure 5 (below). This attempt ran afoul of address typing. The certain region of address space allocated for names and numbers does not include the location of the machine language functions. Attempting to refer to a number outside this atom space had interesting but not highly desirable consequences.

The second possibility that I considered was to use one of the parameters to return a value. A valid muLISP number is passed as the port address. Once its value has been used, it is no longer required. Why not simply amend its vector value to reflect the value of the input port and return the same pointer? Why not indeed?

This suggestion is precluded by the constraint on muLISP machine language functions to be *call by value* functions, which means that the function arguments are evaluated before they are passed to the function. What is passed is a pointer to the value of that argument, a muLISP number that may be shared by many other objects in the LISP environment. Changing the value of this number changes all these values simultaneously. Again the results are interesting but not always desirable.

The final and ultimately successful tack was to discover how muLISP handled the same problem. Some tedious disassembly revealed the existence and location of a subroutine that converts the number in register A into a valid muLISP number. So IN simply jumps to this subroutine with the value it obtains from the port. In the listing this subroutine is labeled RETURN_A.

This method of returning a value to muLISP is simple. However, it has the disadvantage of requiring prior knowledge of the absolute location of the subroutine RETURN, which may differ from version to version of the inter-

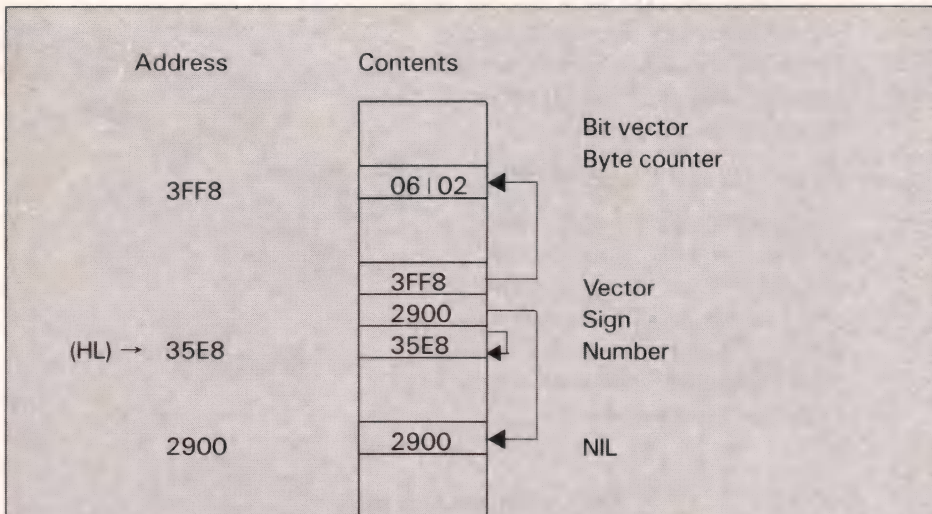


Figure 4

Memory contents are shown as words (16 bits) with the order of the most and least significant bytes reversed, except in the case of the byte counter and the bit vector, which are shown as bytes. The bit vector precedes (has a lower address than) the byte counter. Location 2900H is a special atom: NIL. The sign of a nonnegative number always points to NIL.

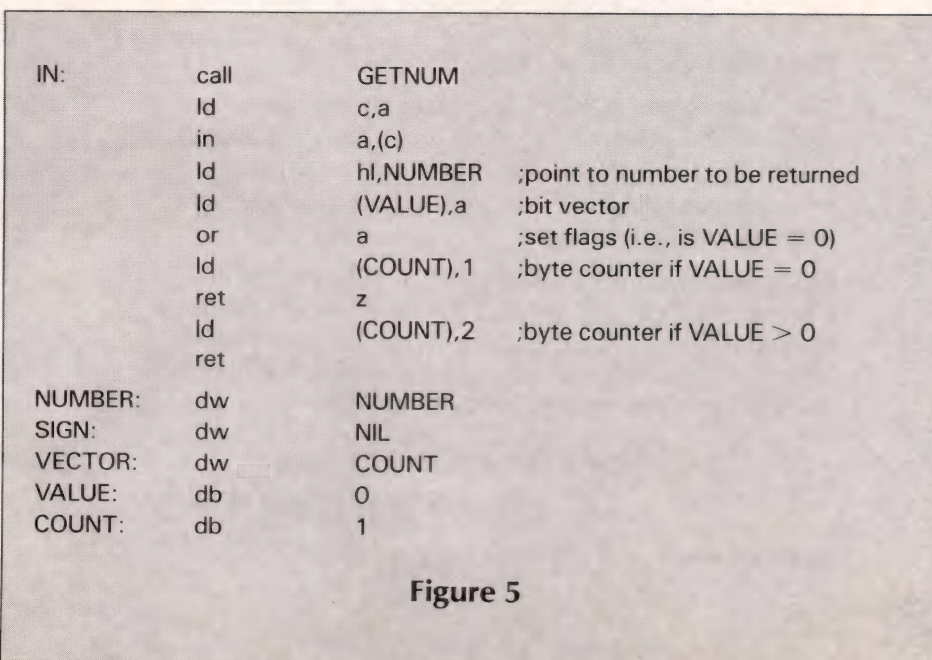


Figure 5

preter. The next section provides some hints on finding the location of RETURN_A.

A final point: Every machine language function must return a value; that is, the HL register pair must point to a bona fide muLISP data object (e.g., NIL). The function OUT simply returns the value of the byte it has output.

Finding RETURN_A

The location of the subroutine RETURN_A can be found by disassembling the function MEMORY, the location of which can be found with the following muLISP command:

(GETD MEMORY)

Remember that the returned address will be in decimal unless you change the radix. In my system the code for MEMORY is as follows (with the radix

changed to 16):

```
12D5 CALL 0650
      JP NC,07BE
      LD A,(HL)
      PUSH AF
      EX DE,HL
      CALL 0672
      JP NC,12E5
      LD (DE),A
      POP AF
      JP 0FC6
```

This last address, 0FC6, is the location of the subroutine RETURN.

Conclusion

Adding some simple I/O functions to muLISP turned out to be more complex than I first envisaged. In the process I learned some interesting details about the internal structure of the interpreter. I pass on the results of my

investigations in the hope that they may be of use to others who want to add other machine language functions to muLISP. The listing gives the requirements for the primitive I/O functions IN and OUT.

My efforts may also be of value to other programmers. The section that deals with obtaining a suitable block of memory for the machine language functions applies quite generally to CP/M programs. Although the remaining sections are more specific to muLISP, the general concepts addressed are worthy of attention by a wider audience. Any LISP implementation is likely to follow a similar structure, while any high-level language is likely to provide similar problems.

DDJ

Reader Ballot

Vote for your favorite feature/article.
Circle Reader Service No. 192.

muLisp Listing (Text begins on page 18)

;Modifications to muLISP to provide primitive I/O functions
;The function and operation of this code is fully explained in the accompanying text

```
.280
aseq

0000'
0005      GOBDOS      equ      5           ;CP/M entry point
0134      START      equ      0134H        ;entry point to muLISP
0FC6      RETURN_A   equ      0FC6H        ;muLISP subroutine
2880      ENDLISP     equ      2880H        ;end of original code
A300      BDOS       equ      0A300H       ;depends on memory size

                                org      100H
0100      C3 2880     jp      INIT          ;muLISP jump table
0103      C3 A203     jp      IN
0106      C3 A20C     jp      OUT
0109      C3 0109     USER3: jp      USER3
010C      C3 010C     USER4: jp      USER4

                                org      ENDLISP

2880      INIT:
2880      21 A200     ld      hl,BDOS - 100H ;patch GoBDOS with lower address
2883      22 0006     ld      (GoBDOS+1),hl ;see Figure 3
2886      EB         ex      de,hl         ;and relocate functions
2887      21 2892     ld      hl,CODE      ;ie. move machine language functions
288A      01 0027     ld      bc,LENGTH    ;up to memory page immediately
288D      ED B0       ldir                ;below BDOS
288F      C3 0134     jp      START        ;of muLISP

2892      CODE      equ      $
```

(Continued on next page)


```

                                .phase BDOS - 100H                ;this is the code of the machine language
                                                                ;functions assembled to reside at BDOS - 100H

A200  C3 A300                  jp      BDOS                    ;jump for BDOS calls

A203  CD A219                  IN:    call   GETNUM                ;get port number
A206  4F                      ld      c,a
A207  ED 78                   in      a,(c)                ;input value at port
A209  C3 0FC6                 jp      RETURN_A                ; return to muLISP

A20C  D5                      OUT:    push   de                ;save output value for return value
A20D  CD A219                  call   GETNUM                ;get port number (pointed to by hl)
A210  4F                      ld      c,a
A211  EB                      ex      de,hl
A212  CD A219                  call   GETNUM                ;get value to output
A215  ED 79                   out    (c),a                ;output value to port
A217  E1                      pop     hl                ;return value (from stack)
A218  C9                      ret

A219                          GETNUM:
                                ; returns a muLISP number mod 256
                                ; on entry, hl points to number

A219  23                      inc     hl                ;skip over first two (16 bit) fields
A21A  23                      inc     hl                ;see Figure 4
A21B  23                      inc     hl
A21C  23                      inc     hl

A21D  7E                      ld      a,(hl)                ;hl now points to pointer to the bit vector
A21E  23                      inc     hl                ;load pointer in hl
A21F  66                      ld      h,(hl)
A220  6F                      ld      l,a

A221  7E                      ld      a,(hl)                ;hl now points to the byte count
A222  3D                      dec     a                ;load byte counter into a
A223  C8                      ret     z                ;return 0 if byte count = 1
                                                ;otherwise

A224  2B                      dec     hl
;point to LSB
A225  7E                      ld      a,(hl)
A226  C9                      ret
;return LSB

                                .dephase

0027                          LENGTH equ    $ - CODE

                                end

```

End Listing

B. G. MICRO

P. O. Box 280298 Dallas, Texas 75228

(214) 271-5546



Big Computer Mfg. Makes \$900,000 Goof!!

COMPUTER/DISK DRIVE SWITCHING POWER SUPPLY

ORIGINAL
OEM COST
\$72 EACH!

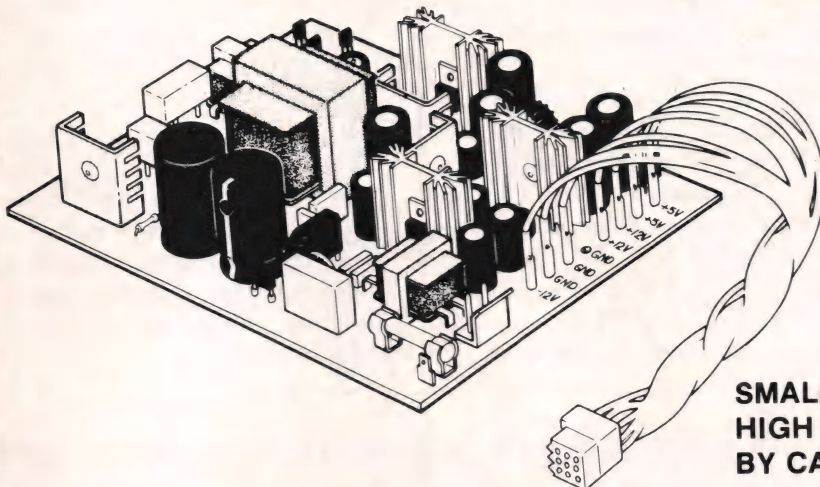
ORIGINALLY DESIGNED TO RUN A Z-80
BASED SINGLE BOARD COMPUTER
WITH TWO 5-1/4 IN. DISK DRIVES AND
CRT MONITOR.

BRAND NEW: UNUSED!

\$37⁵⁰ EA

3 FOR \$95⁰⁰

ADD \$1.50 PER UNIT FOR UPS



SPECS: + 5VDC 5 AMPS MAX
#1 + 12 VDC 2.8 AMPS MAX
#2 + 12 VDC 2.0 AMPS MAX
- 12 VDC .5 AMPS MAX

INPUT: 115 or 230 VAC 60Hz

SMALL SIZE: 6-1/8 x 7-3/8 In.
HIGH EFFICIENCY SWITCHER MFG.
BY CAL. DC IN USA!

The poor Purchasing Agent bought about 10 times as many of these DC switchers as his company would ever use! We were told that even in 10,000 piece lots they paid over \$72 each for these multi-output switchers. When this large computer manufacturer discontinued their Z-80 Computer, guess what the Big Boss found in the back warehouse; several truckloads of unused \$72.00 power supplies. Fortunately we heard about the deal and made the surplus buy of the decade. Even though we bought a huge quantity, please order early to avoid disappointment. Please do not confuse these high quality American made power supplies with the cheap import units sold by others.

TERMS: Orders over \$50 add 85¢ insurance. No COD. Tex. Res. Add 6% Sales Tax. Subject to prior sale. Foreign orders: US funds only. We cannot ship to Mexico. Foreign countries other than Canada add \$6 per board shipping.

Circle no. 6 on reader service card.

Program Monitor Package

Using Interrupts to Instrument Applications

by Alan Bomberger

The optimization of programs for minimum execution time is a passion for some programmers and a necessity for others. I sometimes argue that the three phases of a good programmer's development are: first, optimize for minimum number of instructions; second, optimize for minimum execution time; and last, optimize for readability. Once programmers reach the last phase, they can begin to use computer aids to optimize for execution time and program size. The package described here is one of those aids.

Many quickly written programs find their way into production products; when they do, they often need to be rewritten. Sometimes small, nagging bugs are removed, sometimes features are added, but most often the program that worked well on a little data takes far too much time when given a full measure of data. Unguided attempts to

tation package to determine where the program spent most of its time.

The ideas behind such a package are not new, and the techniques for implementing the package on large systems are well known. I simply applied these ideas and techniques to a CP/M system and developed a Program Monitoring Package (PMP) for CP/M.

Overview

The principal technique for monitoring programs is to examine the program counter (the address of the fetch of the next instruction) while the program is running. When machines had front panels, a rather crude analysis consisted of watching the lights: the lights that glowed the brightest represented the address in memory from which most of the instructions were fetched. This technique, however, could locate only extremely time-consuming sections of code.

"Unguided attempts to optimize programs for speed often result in pointless modifications. I needed to develop an instrumentation package to determine where the program spent most of its time."

optimize programs for speed often result in pointless modifications that only reduce readability of the program.

I recently examined a program of mine (a spelling checker) with an eye to optimizing its speed. I examined the code in places where I thought the program could be "tightened up" but felt that changes in those areas would produce buggy code with no obvious benefit. I needed to develop an instrumen-

A more precise technique is to record the value of the program counter at random instants while the program is running. To record every value of the program counter is usually impractical, so the program is divided into small sections based on the address. A counter is incremented each time the program counter is seen within a section. Coarse samples might divide the program into 1024-byte sections, while fine samples might use sections as small as 32 bytes. Finer sampling implies more counters. Two thousand counters are needed to sample a 64K program with 32-byte sections.

Alan Bomberger, Poor Person Software, 3721 Starr King Circle, Palo Alto, CA 94306.

With enough samples it is possible to identify small sequences of time-consuming instructions. I once analyzed a heavily used program for hours to locate the code that was slowing it down. Coarse sampling isolated a subroutine that was consuming much more than its fair share of computer cycles. Detailed sampling within the subroutine finally isolated a divide instruction as the culprit. A rewrite of the subroutine without the divide instruction resulted in a 10% increase in program speed. In another case, coarse sampling isolated a sort subroutine with a very poor algorithm. A rewrite of the sort made the program run 10 times faster.

Sampling may reveal that no isolated areas of the program are particularly inefficient. In these cases an examination of the algorithms involved is necessary to determine if there are more efficient ways of solving the problem.

Later in the article I will discuss the pitfalls of sampling (random and otherwise) the program counter. It is, however, a good technique for isolating the immediate areas in a program that must be examined in detail. Other techniques also analyze the behavior of a running program. One popular technique is to use a special compiler that inserts subroutine calls to a recording feature at significant places in the program. The inserted subroutine records the events for later analysis. The analysis shows DO LOOP counts, SUBROUTINE counts, and even counts on each branch of an IF statement.

The major advantage of this technique is that it enables a complete flow analysis, with exact identification (in terms of the source language) of time-consuming code. The major disadvantage is that the technique depends on the availability of a special compiler for the language being used. By contrast, the interrupt-driven sampling technique is applicable to any program in any language. Its major disadvantage is that it usually provides no knowledge of the program structure, requiring the programmer to correlate the program counter values with particular sections of code.

Interrupts

Just how does this technique sample the program counter? Most computers, large and small, have a feature

that allows some sort of timer to interrupt the program. As its name implies, an interrupt is an unpredictable change in the status quo. In other words, some program segment in the computer memory is executed out of sequence and without the knowledge or cooperation of the main program.

Timer-based interrupts are the basis for program monitoring. The interrupt executes a program segment that examines the program counter of the interrupted instruction and increments a counter based on the section of the program executing at the time of the inter-

rupt. The segment then restores the status of the machine to its exact state at the time of the interrupt and returns control to the program. The only effect on the program is that it runs a little slower.

Taking Control from BDOS

Because the instructions that the computer executes when an interrupt occurs are neither part of the application program nor CP/M, the programmer must find some technique to reserve a portion of the computer's memory for this purpose. Some systems have mem-

INTRODUCING THE LATEST IN HIGH QUALITY PRODUCTIVITY TOOLS FOR MICROCOMPUTER SOFTWARE DEVELOPERS AND PROGRAMMERS

{SET} Tools -

- Operate on most popular MS-DOS and CP/M systems.
- Can be used with any source language.
- Improve development productivity.
- Provide assistance for the tedious task of maintenance.

{SET:DIFS}™ Source File Comparator \$139.00

- Fast, smart and accurate
- Use for regression testing, too
- Difference display highlighting

A/P/L Options available as *add-ons* to {SET:DIFS} to provide for minimized communications with the ADR or PanValet mainframe librarians or with {SET}'s Batch Line Editor, {SET:LIKE}. \$20.00 per option
{SET:LIKE} add-on to {SET:DIFS}. \$40.00

{SET:GXREF}™ Cross Reference Utility \$79.00

- Supplied parameter files allow use of any source language
- Cross references multiple files at once

{SET:PATCH}™ Object File Editor \$79.00

- Quickly apply changes to any file type
- Hexadecimal and ASCII display and change entry
- Easy to use with cursor and function keys

{SET:SCIL}™ Source Code Interactive Librarian \$349.00

- Maintains history of changes
- Reduces storage by identifying differences from level to level
- Provides control over concurrent development efforts by detecting overlapping changes

PC-Demo Disk and Manual available for \$35.00

{SET} Tools are available individually or, better yet, select a combination of tools to meet your specific needs.

Multiple copy discount available.

{SET} Get {SET} for Success
System Engineering Tools, Inc.
645 Arroyo Drive • San Diego, CA 92103

COD, Check with order, Master Card or VISA accepted.

To order {SET} tools or for more information, call
System Engineering Tools, Inc. (619) 692-9464.

Circle no. 71 on reader service card.

ory available at addresses beyond either the program or CP/M. Such systems, however, are becoming rare, and the programmer is likely to need a method that uses a portion of the memory normally used for programs. Such a method has been part of the CP/M lore for many years.

The instruction at location 5 of a CP/M system is a jump to the CP/M BDOS. Location 6 is the address of the lowest memory used by the BDOS; application programs can use any memory up to this point. When an application is not in control, the CCP (console command processor) is loaded immediately below the BDOS. To reserve a portion of memory for the interrupt program, the programmer must change the address at location 6. Once this is done, applications will not use the memory reserved for the interrupt program and the CCP remains in memory.

It is easy to see that the instruction at location 5 must now jump to the lowest address used by the interrupt program; this must be a jump to the address that was in location 6 before the change was made. This "bounce pass" does not interfere with the operation of CP/M.

Three additional details must be worked out. First, a program must load the interrupt program into the memory reserved for it. An initialization program usually accomplishes this, running as a standard application that moves a block of code from low memory to just below the CCP. This code must be assembled assuming a location below the CCP and not in low memory; assembly language constructs explained in Listing Two (page 32) accomplish this.

Second, an application program should be able to retrieve the samples of the program counter for analysis. The interrupt code adds two new BDOS function calls for his purpose. Because all BDOS calls now pass through the interrupt code on their way to the CP/M BDOS, it is possible to intercept certain calls and process them in the interrupt code itself. Each BDOS request is examined to see if it is one of the new functions. If not, the request passes on to the real BDOS. If it is a new function, the processing takes place in the interrupt code.

In this package one BDOS function

zeros the counters and starts the program monitor. Another function returns a pointer to the counters and an indication of the address range of each counter. When the interrupt code interprets these BDOS functions, return is directly to the calling program without jumping to the real BDOS.

The third action is to modify the interrupt vector in low memory so that the PMP is entered each time a timer interrupt occurs. I will explain this action in detail further on.

Taking Control from BIOS

Several changes in the behavior of the BIOS are necessary. The WARM BOOT function must not overlay location 6 with the real BDOS address; it need not read in the CCP (which is protected from the applications by the value in location 6) or the BDOS. To intercept the BIOS WARM BOOT call, the programmer must change the address in the BIOS jump table.

Location 0 is a JMP to a vector of JMP instructions. The first of these is a jump to WARM BOOT. (Other jumps in this table jump to various I/O routines of the BIOS.) A jump to location 0 therefore causes a jump to the WARM BOOT routine. To intercept the WARM BOOT function, we need only change the first address in the BIOS jump table to point to our code.

To simulate the WARM BOOT, we use a BDOS function RESET DISK: the stack pointer is reset and location 6 is reset to the correct value (which points to our code). The reset of location 6 is necessary because some processors that use a similar technique of modifying location 6 do not reset it when done. DDT is an example.

Two other modifications to the BIOS jump table are necessary. Many systems do not expect interrupts to be enabled during CP/M operation. Interrupts do not bother CP/M, but they may bother the BIOS disk I/O functions: an interrupt during the transfer of data to and from the disk can cause a loss of data. To avoid this, many BIOS systems disable interrupts before performing the disk I/O. Not all of these systems, however, enable interrupts when they are done. Some systems do not bother with interrupts at all, under the assumption that they are never enabled.

For the program monitor to function without disturbing the operation of CP/M, it is necessary to assure that the system disables interrupts before entering the BIOS disk read and write routines and enables interrupts when the disk I/O is complete. We accomplish this by intercepting the jumps to the disk read and disk write functions of the BIOS. By modifying the jump table so that calls to the BIOS disk functions come first to the interrupt code, we can instruct the system to disable interrupts, call the real BIOS function, and then enable interrupts.

It is worth noting here that this technique uses additional stack space. What used to be a simple call to the BIOS becomes two calls: first the interrupt code is called and then the interrupt code calls the BIOS. This requires two extra stack locations. If an application does not have enough stack space, it may not run correctly. Eliminating the use of extra stack space is possible, but it introduces additional complications.

Borrowing Interrupts

The program monitor uses a timer interrupt to sample the program counter. Some CP/M systems already use a timer interrupt for some other purpose. For example, my system uses timers to simulate console interrupts. Others use timers to update clock values and the like. The program monitor works in these environments because it is aware that it may have to share the timer interrupt among several programs.

If a system uses timer interrupts, one of the fixed interrupt locations in low memory (0 - 38) is set up with the address of the timer interrupt routine. The occurrence of a timer interrupt calls this fixed location. The content of the fixed location usually is a jump to the program that handles the timer interrupt. When the interrupt program is complete, it enables interrupts and returns to the running program.

The program monitor uses the intercept technique to borrow the timer interrupt. The programmer changes the fixed location contents to a jump to the program monitor, which executes the code to record the program counter. When the program monitor finishes, it jumps to the old interrupt program instead of returning to the running pro-

gram. The old interrupt program now runs as if it had been jumped to directly, returning to the running program when it is finished.

Again note that this interception may require additional stack space. I have coded the program monitor very carefully to use as little stack space as possible. Registers are saved outside of the stack and calls are minimized.

Before the program monitor can begin, it must determine whether the timer interrupt is in use or not. If not, the program must initialize the hardware that causes the interrupt and enable interrupts. If the timer is in use, the program only modifies the interrupt location to borrow the interrupt.

A test is necessary to determine whether the timer is running or not. This test borrows the timer interrupt and sets a flag if the timer goes off. If after a short wait the flag is on, the timer was previously running. Different techniques initialize the program monitor depending on whether the timer was running or not. All of these techniques are documented in Listing Three (page 44).

Sampling Theory without Mathematics

As the word implies, sampling is an incomplete recording of information. It is subject to a certain class of errors that relate to the relative frequency of the samples and the information that is sampled. Imagine a large program loop that takes 10 msec to complete. Sampling the program counter every 10 msec identifies only one portion of the loop. The complete identification of a loop requires knowing the bounds of the loop as well as its location. (The bounds of most loops can be identified from the program listing. However, determining if some portions of the loop take longer than others requires samples throughout the range of the loop.) If the same program loop were to take only 2 msec, samples every 10 msec might not identify it at all. Increasing the sample rate to every 1 msec would identify either loop correctly.

Most programs are a series of loops within loops and represent periodic processes. As seen above, periodic sampling of periodic processes can provide misleading information. A sufficiently high sampling rate will ensure

that information is not lost but may require recording an excessive amount of information. Sampling at random intervals takes advantage of the benefits of a high sampling rate while limiting the amount of information recorded.

When sampling a 10 msec loop at 1 msec intervals, little information is lost if some of the samples are discarded. The sample rate of 1 msec ensures that loops as short as 2 msec are correctly identified, and discarding samples at random ensures that no information is lost over the long run.

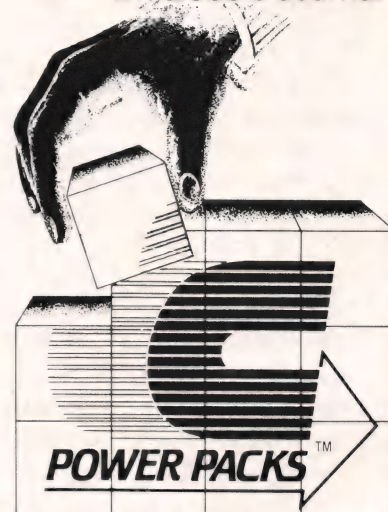
Sampling at a fixed interval of 1 msec and discarding random samples is equivalent to sampling at random intervals that are a multiple of 1 msec. The minimum interval of the randomizing function determines the accuracy of the data based on periodic sampling. Random sampling or randomly discarding periodic samples does not increase the accuracy of the information but decreases the volume of data that must be recorded.

Random discarding of periodic samples is a method of simulating random sampling that can be implemented in software. This simulation is often necessary because many small computers have timers with intervals fixed by wiring changes—these timers cannot be set at random intervals by software. The software simulation of random sampling is not without significant cost. It involves fielding a timer interrupt at the highest periodic rate and using a counter set at a random value to determine which samples are recorded. The software to save and restore status at the interrupt and to calculate the random value for the counter may involve as many as 100 instructions. A Z80 with a 4 MHz clock executes between 200 and 300 instructions per msec, so the effect of the sampling overhead under these conditions is a 30–50% decrease in speed.

Absolutely accurate data is unnecessary as long as the possible errors in the data are understood. With this understanding and with a desire to minimize the overhead involved in sampling, the PMP as implemented on my hardware uses periodic sampling with an interval of 3.3 msec (which is, conveniently, the shortest interval available with my NorthStar Horizon computer). This works out to about 300

"This is a beautifully documented, incredibly comprehensive set of C Function Libraries."

— Dr. Dobb's Journal



COMPLETE SOURCES

- **PACK 1: Building Blocks I** \$149
250 Functions: DOS, Printer, Video, Asynch
- **PACK 2: Database** \$399
100 Functions: B-Trees, Variable Records
- **PACK 3: Communications** \$149
135 Functions: Smart-modem™, Xon/Xoff, Modem-7, X-Modem
- **PACK 4: Building Blocks II** \$149
100 Functions: Dates, Text Windows, Pull-down Menus, Data Compression
- **PACK 5: Mathematics I** \$99
35 Functions: Log, Trig, Square Root
- **PACK 6: Utilities I** \$99
Archive, Diff, Replace, Scan, Wipe (Executable Files only)

Lattice™, Microsoft™, DeSmet™, C1-86™ Compilers on IBM PC/XT/AT™
Small and Large Memory Models.
Credit cards accepted
(\$7.00 handling/Mass. add 5%)



165 Bedford Street
Burlington, Mass. 01803
(617) 273-4711

NOVUM ORGANUM

Circle no. 82 on reader service card.

29

samples per sec for a Z80 with a 4 MHz clock or about 1 sample every 1000 – 2000 instructions.

Because loops of less than 1000 instructions are common in programs coded in assembly language, the accuracy of sampling at this rate is not sufficient to rely totally on the output of the PMP when determining which areas of code to examine. The fact that samples represent a range of addresses rather than a specific instruction mitigates this problem. In a program divided into 64-byte ranges (1024 counters for a full 64K memory), 10 counters would represent a 300-instruction loop. The only way to establish the exact location of code inefficiencies, given this information, is to examine the code. The output of the PMP, however, is helpful in isolating areas that need detailed examination.

Analyzing Results

The first decision we must make when using the PMP is the size of the program section represented by each counter. Because this currently is determined at assembly time, we may need several versions of the PMP for the complete study of a program. The PMP allows a minimum program section size of 32 bytes, which requires 2048 counters occupying 4K of memory. If

the program is small (thus requiring the 32-byte section size), there will be adequate memory for the counters.

The second decision is the selection of the sample rate. Most programs chosen for analysis with the purpose of increasing their efficiency will be long-running ones. The selection of the sample rate must achieve a balance between identifying small loops and keeping the counters from overflowing. In my case, the NorthStar hardware limits the sample rate choices.

Once we have configured the hardware and software for the PMP, the analysis of a program consists of four steps: The PMP must be loaded into the CP/M system, the program monitor started, the program run, and the monitor stopped and results printed. The system I use prints a matrix of counter values for each program section.

We then examine the program listing and identify each section of the program with a higher-than-average counter value. Be sure to note when only one portion of a loop has a high counter value. This usually indicates that the loop interval is close to the sampling interval, and we must be careful in drawing conclusions about the changes that must be made in that part of the program. It might help to

add a long string of NOPs to such loops and rerun the program with the monitor. Adding the NOPs (or statements like $A = A;$) will change the length of the loop and increase the accuracy of the sample information.

Conclusions

I used the PMP to analyze the Poor Person's Spelling Checker after I had converted it (originally written for Z80) to 8080 and had made several enhancements (increasing the degree of compaction in the lexicon and adding hash chains to the word lists). See Listing One (below). I had expected the expansion and hashing routines to be a significant factor in the performance. What I discovered was that some sloppy code in a routine to follow chain links was responsible for most of the performance problem. I spotted other minor efficiency problems and reworked the code. The routine to follow chain links is still a major consumer of CPU resources, leaving me to conclude that further improvements will come only with redesign.

DDJ

Reader Ballot

Vote for your favorite feature/article.
Circle Reader Service No. 193.

Program Monitor Package (Text begins on page 26)

Listing One

An example of the PMP in use.

The Poor Person's Spelling Checker was monitored using 32-byte program sections while it was checking the spelling of this article. The sections of memory that were not used have been edited out of the listing to save space. The samples at high addresses are from the BDOS and BIOS.

```
A>pmp on
A>monitor -b
A>b:spell pmp.ddj pmp.mis
```

Poor Person's Speller (c) 1981,1983 Alan Bomberger

```
0715  distinct words in text
Enter lexicon file name (.LEX assumed) or <RETURN>
b:spell
```

```
Begin spelling check pass
Enter lexicon file name (.LEX assumed) or <RETURN>
```

TM
BM

LM
 RM
 DS
 CTR
 S
 READABILITY
 UNGUIDED
 POINTLESS
 INSTRUMENTATION
 CP
 M
 MONITORING
 PMP
 OVERVIEW
 LIGHTS
 INSTANTS
 K
 SUBROUTINE
 NON
 PREDICTABLE
 QUO
 BDOS
 BECOMMING
 CCP
 BIOS
 JMP
 O
 DDT
 MILLISECONDS
 MILLISECOND
 RANDOMIZING
 SOFTWARE
 INTERURRUPTS
 SLOWDOWN
 THUS
 INDENTIFYING
 NOP

A>monitor -e

	0	20	40	60	80	A0	C0	E0	100	120	140	160	180	1A0	1C0	1E0
0	3								14	D			2			6
200	1D6	18B	36	57	25A	2D5	AD	1A	3		45	C2				
400	8F	324	224	A0	391	22C	35C	2AB	F2	4	6	2	3	C0	31E	9
600		1														
800													1			
A00																
C00																
E00																
1000																
1200																
1400																
1600																
1800																
1A00																
1C00																
1E00																

.

(Continued on next page)

Program Monitor Package (Listing Continued, text begins on page 26)

Listing One

```

0 20 40 60 80 A0 C0 E0 100 120 140 160 180 1A0 1C0 1E0
C000
C200
C400
C600
C800
CA00
CC00
CE00
D000      3   F   1   3           5  10   6
D200      6D   A   A   1   9   B   8   D  16   5   E  11   9  30
D400      4      1   1   7   2   1   4   4   C   D  20  11   1   1   2
D600      6      9
D800
DA00      1   2   5
DC00      3   7           3   2   B   F   9           2
DE00      42D1547           2

0 20 40 60 80 A0 C0 E0 100 120 140 160 180 1A0 1C0 1E0
E000      3   B   9   A
E200
E400
E600      B38 E73
E800
EA00
EC00
EE00
F000
F200
F400
F600
F800
FA00
FC00
FE00

```

```

0 20 40 60 80 A0 C0 E0 100 120 140 160 180 1A0 1C0 1E0
A>

```

End Listing One

Listing Two

Assembly Listing of the Program Monitor

Copyright © 1984 by Poor Person Software. All rights reserved.
Permission is granted for personal, non-commercial use only.

```

false      aseq                ; absolute assembly
true       equ      0
           equ      not false
;
; the following four equates must be changed for
; the particular hardware configuration of your system
;
ccp        equ      0c700h      ; ccp base
intvct     equ      4           ; the S100 interrupt number of the clock
enable     equ      true        ; enable interrupts when done
nbits      equ      11         ; number of bits in counter address
;

```



```

;      11 bits means every 32 bytes
;      10 bits means every 64 bytes
;      9 bits means every 128 bytes
;      8 bits means every 256 bytes
;
bdos    equ    5           ; jump for bdos
boot    equ    0           ; warm boot place
space   equ    2 shl (nbits) + 0200h
pmpbas  equ    ccp-space   ; base of pmp when active
setdma  equ    26          ; bdos equate for setdma
rstdsk  equ    13          ; bdos equate for reset disk system
pstring equ    9           ; bdos equate for print string
fcb     equ    05ch        ; fcb base (for input options)
intloc  equ    intvct*8    ; location of jmp for interrupt
;
;      org    0100h
;
;      lda    fcb+2         ; f or n (part of "off" or "on")
;      cpi    'N'           ; on
;      jz     onpmp
;      cpi    'F'           ; off
;      jz     offpmp
;      ret                ; forget it
onpmp:
;
;      see if already on
;
;      lxi    h,intrcpt     ; my bdos jump address
;      lda    bdos+1        ; compare with bdos jump that is active
;      cmp    l
;      jnz    onit          ; it is not mine so we can take over
;      lda    bdos+2
;      cmp    h
;      rz                ; it is already on
onit:
;
;      move the code into high memory
;
;      lxi    h,pmpbas      ; destination of code
;      lxi    d,pmp         ; assembled code
;      lxi    b,0200h       ; move this many
10:
;      ldax   d             ; get byte
;      mov    m,a           ; put byte
;      inx    h
;      inx    d
;      dcx    b
;      mov    a,b
;      ora    a
;      jnz    10
;      mov    a,c
;      ora    a
;      jnz    10           ; move all bytes
;
;      now modify the existing addresses in 1 and 6
;
;      di                ; we must not be interrupted
;      lhld   bdos+1       ; current bdos jump
;      shld   jmpbd+1      ; save bdos jump so can restore
;      lxi    h,intrcpt    ; get new jump to bdos
;      shld   bdos+1       ; replace bdos jump so can intercept them
;
;      copy the disk error addresses
;
;      mvi    b,8           ; this many bytes
;      lhld   jmpbd+1
;      xchg

```

(Continued on next page)

Program Monitor Package (Listing Continued, text begins on page 26)

Listing Two

```

        lhld    bdos+1        ; new bdos address
        mvi     e,9
        mvi     1,9          ; offset of address list
c11:
        ldax    d
        mov     m,a
        inx     h
        inx     d
        dcr     b
        jnz     c11          ; copy all 8 bytes
;
; intercept disk read/write routines to our place
; so we can disable the clock in case the BIOS doesn't
; mask interrupts when it needs to
;
        lhld    1            ; bios vector base
        lxi     b,37         ; offset for read
        dad     b            ; to 2 byte address of read
        mov     e,m
        inx     h
        mov     d,m          ; get both bytes
        xchg
        shld    biores       ; to our call
        xchg
        lxi     d,disred
        mov     m,d
        dcx     h
        mov     m,e          ; put in our address
;
        lhld    1            ; vector base
        lxi     b,40         ; to 2 byte address for write
        dad     b
        mov     e,m
        inx     h
        mov     d,m          ; to de
        xchg
        shld    biowrt       ; to our call
        xchg
        lxi     d,diswrt
        mov     m,d
        dcx     h
        mov     m,e          ; put ours into vector
;
; now do the same for warm boot
;
        lhld    1            ; address of warm boot jump
        inx     h            ; low byte
        mov     e,m
        inx     h
        mov     d,m          ; get it
        xchg
        shld    oldjpo       ; save it
        xchg
        lxi     d,jmp0       ; our address
        mov     m,d
        dcx     h
        mov     m,e
;
        if      enable
        ei
        endif

```

(Continued on page 36)

UNIX

STEP UP TO ELEGANCE
**UNIX is the environment professionals depend on for productivity, portability, flexibility and simply making programming a pleasure.*

THE SOLUTION™

- **EXPANSIVE SOFTWARE DEVELOPMENT FACILITIES** Language and Operating System design.
- **LANGUAGES INCLUDE:** C, Fortran 77, RATFOR, COBOL, SNOBOL, BS, Assembler + LISP.
- **USENET Bulletin Board System**—800 + networked international UNIX sites, 190 + categories, 300 + new articles per day.
- **Inter/intra system mail + communications.**
- **UNIFY:** for professional data-base application development.
- **UNIX & System enhancements** from U.C. Berkeley and Kormsmever Electronic Design Inc.
- **Online UNIX manuals + Expert consultation** available.
- **Just a modem call** from 300 + cities nationwide via Telenet.
- **FAST and LOW COST,** low as \$8.95 hr. connect.
- **\$24.95 = 1 hr. FREE system time,** SOLUTION News, BYTE BOOK: Introducing The UNIX System, 556 pp. *UNIX is a trademark of Bell Labs.

Kormsmever
 ELECTRONIC DESIGN, INC.

5701 Prescott Avenue
 Lincoln, NE 68506-5155
 402/483-2238 10a-7p Central



Payment via VISA or MasterCard

Circle no. 39 on reader service card.

Introducing the Creative Genius...

YOU. Discover how easy programming can be with DataBurst™.

A unique runtime screen processor and source program generator, DataBurst™ will decrease your program development time and increase the value of your application programs. The unique DataBurst™ screen editor provides fast, easy screen design. Program independent screen formats reduce both development and maintenance time.

During execution of your program, DataBurst™ controls all user interaction through one assembly language interrupt service routine, requiring as little as 14K of memory. A true full-screen processor, DataBurst™ allows unlimited design complexity, and brings a mainframe advantage to your IBM® PC.

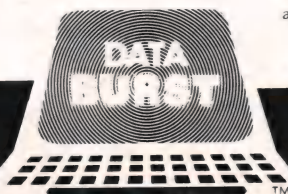
DataBurst™ is available through your local computer retailer or directly from Key Solutions, Inc. To order directly, please send check or money order for \$225* to Key Solutions, Inc., P.O. Box 2297, Santa Clara, CA 95055. Additional language support (BASIC Compiler and C Compiler) is available for \$40* (Please inquire about release dates for other language interfaces).

IBM® is a registered trademark of IBM Corporation.
 DataBurst™ is a trademark of Key Solutions, Inc.

*In California add applicable sales tax.

© Copyright 1984 Key Solutions, Inc.

The Design Tool for the Creative Programmer



KEY SOLUTIONS™

Circle no. 38 on reader service card.

33 KFLOPS

Use your IBM PC (or compatible) to multiply two 128 by 128 matrices at the rate of 33 thousand floating-point operations per second (kflops)! Calculate the mean and standard deviation of 16,384 points of single precision (4 byte) floating-point data in 1.4 seconds (35 kflops). Perform the fast Fourier transform on 1024 points of real data in 6.5 seconds. Near PDP-11/70 performance when running the compute intensive Owen benchmark.

WL FORTH-79

FORTH-79 by WL Computer Systems is a powerful and comprehensive programming system which runs on the IBM PC (and some compatibles). If your computer has the 8087 numeric data processing chip (NDP) installed, then this version of FORTH-79 will unleash the awesome floating-point processing power which is present in your system. If you haven't gotten around to installing the 8087 NDP coprocessor in your computer, you can still use WL FORTH to write applications using standard FORTH-79.

System includes editor, memory dump, decompiler, nondestructive stack print-out, screen printer and screen copy utilities. FORTH sources for these utilities are included.

Unlike most other products, the **complete** source is available at a very affordable price.

Package 1 includes FORTH-79 versions with and without 8087 support. Included are screen utilities, 8087 and 8088 FORTH assemblers. \$100

Package 2 includes package 1 plus the assembly language source for the WL FORTH-79 nucleus. \$150

Package 3 includes package 2 plus the WL FORTH-79 source screens used to add the 8087 features to the vocabulary. \$200

Starting FORTH book. \$22

WL Computer Systems
 1910 Newman Road
 W. Lafayette, IN 47906
 (317) 743-8484

Visa and Master Card accepted.

IBM is a trademark of International Business Machines

Circle no. 78 on reader service card.

Program Monitor Package

(Listing Continued, text begins on page 26)

Listing Two

```
;
    ret
offpmp:
;
;    see if pmp is active
;
    lxi    h,intrcpt    ; my bdos intercept address
    lda    bdos+1
    cmp    l            ; see if we are on
    rnz    ; no return quickly
    lda    bdos+2
    cmp    h
    rnz    ; it ain't us
;
    di
    lda    active        ; is it now active
    ora    a
    jz     notact        ; no leave interrupt alone
    lhld   oldtjp+1      ; restore old timer routine
    shld   intloc+1      ; if intloc was not a jump it wasnt meaningful
notact:
    lhld   jmpbd+1        ; restore old bdos jump address
    shld   bdos+1
;
;    must restore bios vectors
;
    lhld   oldjpo        ; warm boot
    xchg
    lhld   1
    inx    h            ; to correct place
    mov    m,e          ; low byte
    inx    h
    mov    m,d          ; high byte
;
    lhld   biored        ; disk read
    xchg
    lhld   1
    lxi    b,37
    dad    b
    mov    m,e
    inx    h
    mov    m,d          ; put back
;
    lhld   biowrt
    xchg
    lhld   1
    lxi    b,40
    dad    b
    mov    m,e
    inx    h
    mov    m,d
;
    lda    active        ; were we on
    ora    a
    jz     pcount        ; no don't fool with clock
    lda    clockon       ; see if clock was on to start
    ora    a
    jnz    pcount        ; yes (leave clock on)
    call   disclk        ; disarm clock
pcount:
```



```

    if      enable
    ei
    endif
    ret
;
;
ds      64

stack:
;
;   from here on all labels must be of the form
;
;   L   EQU      $-PMP+PMPBAS
;
;   to cause the assembler to generate code that can be moved
;   to the location in memory (PMPBAS) where it will reside
;   when it is run.  There are other ways to do this with
;   other assemblers
;
pmp      equ      $           ; moved to high memory
;
intrcpt  equ      $-pmp+pmpbas
;
;   must trap the reset system call
;
;
;   jmp      start
;   ds      14
start    equ      $-pmp+pmpbas
;   mov      a,c             ; see if system reset
;   ora      a
;   jz       jmp0            ; yes simulate warm boot
;   cpi      252             ; how about on
;   jz       startc
;   cpi      251             ; how about off
;   jz       stopc
;   jmpbd    equ      $-pmp+pmpbas
;   jmp      0               ; filled in with "real" bdos address
startc   equ      $-pmp+pmpbas
;
;   zero counters
;
;   lda      active
;   ora      a               ; is it active
;   lxi      h,0             ; if doesn't work set "false"
;   rnz
;   yes cant start the started
;
bytes    equ      space-0200h
;   lxi      b,bytes
;   lxi      d,pccnt
12       equ      $-pmp+pmpbas
;   xra      a               ; get a zero
;   stax     d               ; put a zero
;   inx      d
;   dcx      b
;   mov      a,b             ; check high byte
;   ora      a
;   jnz      12
;   mov      a,c             ; check low byte
;   ora      a
;   jnz      12
;
;   now must "borrow" the timer interupt.  It may be on
;   or it may not be.  must determine the state of timer interrupts
;
;
;   xra      a
;   sta      clockon         ; clear flag
;   lhld     intloc+1        ; possible timer addr
;   shld     oldtjp+1        ; save old address (if active must be jmp)
;   mvi      a,jmp          ; set up my interrupt
;   sta      intloc

```

(Continued on next page)

Program Monitor Package (Listing Continued, text begins on page 26)

Listing Two

38


```

        jnz      stopna      ; it was on leave it on
        call     disclk      ; off the clock
;
stopna equ      $-pmp+pmpbas
;
        if      enable
        ei
        endif
;
        xra      a
        sta      active
        ret
clockon equ      $-pmp+pmpbas
        db      0           ; start with clock off
active equ      $-pmp+pmpbas
        db      0           ; 1 if PMP active
;
;       read sector routine must disable clock interrupts here
;
disred equ      $-pmp+pmpbas
        push     psw
        call     disclk      ; disarm clock
        pop      psw
biored equ      $-pmp+1+pmpbas
        call     $           ; bios read sector filled in
        push     psw
        call     armclk      ; arm clock
        pop      psw
        ret
;
;       write sector routine must disable clock interrupts here
;
diswrt equ      $-pmp+pmpbas
        push     psw
        call     disclk      ; disarm clock
        pop      psw
biowrt equ      $-pmp+1+pmpbas
        call     $           ; bios write sector filled in
        push     psw
        call     armclk      ; arm clock
        pop      psw
        ret
;
;       put your arm clock routine here
;
armclk equ      $-pmp+pmpbas
        lda      active
        ora      a
        rz              ; not active is nop
        mvi      a,050h      ; reset clock flag
        out      6
        mvi      a,0c0h      ; arm interrupts
        out      6
        ei
        ret
;
;       put your disarm clock routine here
;
disclk equ      $-pmp+pmpbas
        lda      active
        ora      a
        rz              ; not active is nop
        mvi      a,040h
        out      6
        ret
;
jmp0 equ      $-pmp+pmpbas

```

(Continued on next page)

Program Monitor Package (Listing Continued, text begins on page 26)

Listing Two

Listing Two

(Continued on page 42)

WIZARD C

Fast compiles, fast code and great diagnostics make Wizard C unbeatable on MSDOS. Discover the powers of Wizard C:

- ALL UNIX SYSTEM III LANGUAGE FEATURES.
- UP TO A MEGABYTE OF CODE OR DATA.
- SUPPORT FOR 8087 AND 80186.
- FULL LIBRARY SOURCE CODE, OVER 200 FUNCTIONS.
- CROSS-FILE CHECKS OF PARAMETER PASSING.
- USES MSDOS LINK OR PLINK-86.
- CAN CALL OR BE CALLED BY PASCAL ROUTINES.
- IN-LINE ASSEMBLY LANGUAGE.
- 240 PAGE MANUAL WITH INDEX.
- NO LICENSE FEE FOR COMPILED PROGRAMS.

The new standard for C Compilers on MSDOS!

Only \$450

WSS

For more information call (617) 641-2379
Wizard Systems Software, Inc.
11 Willow Ct., Arlington, MA 02174
Visa/Mastercard accepted

Circle no. 77 on reader service card.

C UTILITY LIBRARY

The C UTILITY LIBRARY is a set of 200+ functions designed specifically for the PC software developer. Use of the Library will speed up your development efforts and improve the quality of your work.

- BEST SCREEN HANDLING AVAILABLE
- WINDOW MANAGEMENT, COLOR GRAPHICS
- DOS 2 DIRECTORIES, COMMUNICATIONS
- KEYBOARD, PRINTER, TIME/DATE
- EXECUTE PROGRAMS, BATCH FILES
- STRINGS, BIOS, AND MUCH MORE
- ALL SOURCE INCLUDED—NO ROYALTIES

Available for Microsoft/Lattice \$149, Computer Innovations \$149, Mark Williams \$149, DeSmet \$99. Add \$3 shipping. N.J. residents add 6% sales tax. Visa, MC, checks—10 days to clear.

Order direct or through your dealer. Dealer/Distributor inquiries welcome.

ESSENTIAL SOFTWARE, INC.
(914) 762-6605
P.O. Box 1003
Maplewood, N.J. 07040

Circle no. 26 on reader service card.

Six Times Faster!

Super Fast Z80 Assembly Language Development Package

Z80ASM

- Complete Zilog Mnemonic set
- Full Macro facility
- Plain English error messages
- One or two pass operation
- Over 6000 lines/minute
- Supports nested INCLUDE files
- Allows external bytes, words, and expressions (EXT1 * EXT2)
- Labels significant to 16 characters even on externals (SLR Format Only)
- Integral cross-reference
- Upper/lower case optionally significant
- Conditional assembly
- Assemble code for execution at another address (PHASE & DEPHASE)
- Generates COM, HEX, or REL files
- COM files may start at other than 100H
- REL files may be in Microsoft format or SLR format
- Separate PROG, DATA & COMMON address spaces
- Accepts symbol definitions from the console
- Flexible listing facility includes TIME and DATE in listing (CP/M Plus Only)

SLRINK

- Links any combination of SLR format and Microsoft format REL files
- One or two pass operation allows output files up to 64K
- Generates HEX or COM files
- User may specify PROG, DATA, and COMMON loading addresses
- COM may start at other than 100H
- HEX files do not fill empty address space.
- Generate inter-module cross-reference and load map
- Save symbol table to disk in REL format for use in overlay generation
- Declare entry points from console
- The FASTEST Microsoft Compatible Linker available

**SPEED!
FASTER!
SPEED!**

- Complete Package Includes: Z80ASM, SLRINK, SLRIB - Librarian and Manual for just \$199.99. Manual only, \$30.
- Most formats available for Z80 CP/M, CDOS, & TURBODOS
- Terms: add \$3 shipping US, others \$7. PA add 6% sales tax

For more information or to order, call:

1-800-833-3061

In PA, (412) 282-0864

Or write: SLR SYSTEMS

1622 North Main Street, Butler, Pennsylvania 16001

SLR Systems

Circle no. 64 on reader service card.

Program Monitor Package (Listing Continued, text begins on page 26)

Listing Two

```

;      form counter number as high order n bits of the pc address
;
;      lda      pcint          ; get low order interrupt address
;      mov      c,a            ; to c
;      lda      pcint+1        ; get high order address byte
;      mov      b,a            ; to b
;      ora      a              ; clear carry
;
;      form high n bits of address via a double shift right
;      by 15-n (this leaves address * 2 to address 2 byte
;      counters
;
;      shiftc equ      15-nbits
;
;                                     ; 4 for 11 bits (32 bytes)
;      lxi      h,shiftc        ; 5 for 10 bits (64 bytes)
;                                     ; 6 for 9 bits (128 bytes)
;      shiftl equ      $-pmp+pmpbas
;      mov      a,b              ; high byte
;      rar      a                ; low bit to carry 0 to high bit
;      mov      b,a              ; put it back
;      mov      a,c              ; low byte
;      rar      a                ; bring in bit from high, low to carry
;      ani      0feh             ; zap low bit (times 2)
;      mov      c,a              ; put it back
;      dcr      l
;      jnz      shiftl          ; do it n times
;
;      lxi      h,pccnt          ; get base of counters
;      dad      b                ; form the counter address
;      mov      c,m              ; get low byte
;      inx      h
;      mov      b,m              ; get high byte
;      inx      b                ; bump counter
;      mov      m,b              ; put it back
;      dcx      h
;      mov      m,c              ; both back
;      pop      psw
;      lhld     savebc
;      mov      b,h
;      mov      c,l
;      lhld     savehl           ; restore h
;      jmp      retmyt           ; go to next processor or return
;
;      savehl equ      $-pmp+pmpbas
;      ds      2
;      savebc equ      $-pmp+pmpbas
;      ds      2
;      pcint  equ      $-pmp+pmpbas
;      ds      2
;      oldjp0 equ      $-pmp+pmpbas
;      ds      2
;
;      counters
;
;      nc      equ      1 shl (nbits)
;      dw      nc
;      pccnt   equ      $-pmp+pmpbas
;      end

```

End Listing Two

Listing Three

The C program used to control the Program Monitor and to format the counters.

Copyright © 1984 by Poor Person Software. All rights reserved.
Permission is granted for personal, non-commercial use only.

```

/*      This program interfaces with the Program Monitor Package
        to begin sampling, stop sampling, and format counter
        values for printing. The PMP intercepts BDOS calls
        #251, reset counters and begin sampling, and #252
        stop sampling and return a pointer to sample counters
*/

#include bdscio.h

char *fname;           /* pointer to file name */
char iobuf[BUFSIZ];    /* I/O buffer area */

main(argc, argv)
    int argc;           /* number of command tokens */
    char **argv;        /* list of pointers to tokens */
{
    int bdos();          /* declare function */
    int *ptr;            /* declare a pointer */
    char str[5];         /* a string for conversions */
    int ncounter, addr, lctr, addrinc, i, iarg; /* declare integers */
    char *optname;       /* pointer to option */

    if (argc < 2) {      /* first token is command */

```

(Continued on next page)

**THE MOST EXTENSIVE
LIBRARIES ANYWHERE**
for the IBM PC, XT

THE GREENLEAF FUNCTIONS
Total Access to the PC

PROFESSIONAL PROGRAMMER'S TOOLS, With Comprehensive
Manual, Full Source, Examples, Demos, & Newsletter.

Supports All Major Compilers, All Memory Models, DOS 1.1, 2.0, 2.1

IBM • Honeywell • Control Data • GE • Lotus . . .
Hospitals • Universities • Government • Aerospace . . .
Why not for your application?

Save Months of Painstaking Research

GENERAL LIBRARY – What You Would Write If You Had The Time
• 200 functions • DOS 2.0 • Video • String • Printer • Async • Color • Time & Date
• Function Keys • Diagnostic • More

#L2 for Lattice or Microsoft, or #C2 for C1 C86 . . . \$175⁰⁰
#D1 for DeSmet C . . . \$ 99⁰⁰

ASYNCCOMM LIBRARY – Interrupt Driven Communications Capability
• Polled Mode • Interrupt Mode • Ring Buffered • Modem Control • CRC
• XMODEM Protocol Pieces • More

#L3 for Lattice or Microsoft, or #C3 for C1 C86 . . . \$160⁰⁰
#D3 for DeSmet C . . . \$ 85⁰⁰

#CC1 Computer Innovations C86 Compiler V2.1 . . . \$349⁰⁰
#CC2 Lattice C Compiler V2.1 . . . \$395⁰⁰



(214) 446-8641 • Add \$7.00 Shipping Per Item
• Order Direct or Ask Your Dealer • Specify Compiler and
• MasterCard or VISA Accepted • Our Part Number

GREENLEAF SOFTWARE, INC.
2101 HICKORY DRIVE • CARROLLTON, TEXAS 75006

GGM — FORTH™ has HELP* for Z80¹ using CP/M²

GGM—FORTH, a complete software system for real-time measurement and control, runs on any Z80 computer under CP/M using an extended fig-FORTH vocabulary.

GGM—FORTH features:

- Open multiple CP/M files, in any combination of direct-access and sequential-access, fully compatible with all CP/M utilities
- Char. in/out uses CP/M console, lister, file, or port
- On-line HELP* provides instant access to definitions in the run-time GGM—FORTH dictionary
- HELP* file is easily extended to include user definitions using HELP* utility
- HELP* is available during full-screen editing

Complete system and manuals \$150.
Manuals only: \$ 20.
Introductory System: \$ 35.

GGM SYSTEMS, INC. (617) 662-0550
135 Summer Ave., Reading, MA 01867

¹Z80 is a trademark of Zilog, Inc.

²CP/M is a trademark of Digital Research, Inc.

Circle no. 30 on reader service card.

Circle no. 29 on reader service card.

Program Monitor Package (Listing Continued, text begins on page 26)

Listing Three

```

/*      if no arguments (only command token) print help information  */
    printf("Usage:\n\nMONITOR fname -options");
    printf("\n\noptions:\n\n");
    printf("    -B => begin  (fname ignored)\n");
    printf("    -E => end   (Output to fname if present)\n");
    exit();
}

fname="";                                /* initialize pointer to null string */

for (iarg = 1; iarg < argc; ++iarg) {    /* process arguments one at a time in this loop */

    if(*argv[iarg] == '-') {              /* first character of token is '-' */
                                        /* must be an 'option' */
        optname = argv[iarg] + 1;        /* set pointer to option (passed)- */

        if (!strcmp(optname,"B") || !strcmp(optname,"BEGIN")) {

            /* look for B or Begin */
            if (bdos(252,0) == 0) {        /* call PMP to start samples */
                puts("Program monitor not installed or already active\n");
                exit();
            }

        }

        else if (!strcmp(optname,"E") || !strcmp(optname,"END")) {

            /* option is E or END.  print results.  Open optional file for results. */

            if (*fname)                    /* a file has been specified */

                if (fcreat(fname,iobuf) == ERROR) {
                    puts("File create error on %s\n",fname);
                    exit();                /* disk must be full! */
                }

            addr=0;                        /* initialize address value for each line */
            ptr=bdos(251,0);               /* stop sampling and get pointer to counters */
            if(ptr == 0) {                  /* if zero function not present */
                printf("Program monitor not active\n");
                if(*fname) fclose(iobuf);
                exit();                    /* exit */
            }

            ncounter=*(--ptr);              /* number of counters is in front of the counters */
            addrinc=ncounter/16;            /* 16 counters per line */
            addrinc=(16384/addrinc)*4;      /* addrinc is now number of lines same as 65536/addrinc */
            lctr=0;                         /* addrinc is address range covered by each line */
            ptr++;                          /* initialize the line counter */
            while (ncounter-- ) {           /* back to first counter value */
                /* loop for each counter */

                /* every 16 lines give a header */

```



```

if ((lctr % 256) == 0) header(addrinc);

/* print the address at the beginning */
/* of each line */

if ((lctr++ % 16) == 0) {
    if (addr) htos(addr, str); /* convert address to */
    else strcpy(str, " 0"); /* hex, or special case */

    /* if file specified output to it */
    /* else put to console */

    if (!*fname) printf("\n%s ", str); /* no file */
    else fprintf(iobuf, "\n%s ", str); /* file */
    addr += addrinc; /* bump address value */
}
htos(*ptr++, str); /* convert counter to hex */
if (!*fname) puts(str); /* no file put to console */
else fputs(str, iobuf); /* file, put to file */
} /* loop all counters */
header(addrinc); /* trailing header */
if (*fname) { /* if file then close */
    putc('\032', iobuf); /* properly with Z */
    fclose(iobuf);
}
}

/* unknown option */
else printf("Invalid option %s\n", optname);

else fname = argv[iarg]; /* not an option, set file */
/* name pointer */

/*
put out a header
the address increment for each line is divided by 16
to set the address increment for each counter
*/

header(addrinc)
int addrinc;
{
    int colinc, coladdr, i; /* some integers */
    char str[5]; /* a string for conversions */

    colinc = addrinc / 16; /* address increment for each column */
    coladdr = colinc; /* address over second column */
    if (!*fname) puts("\n 0"); /* first column to console */
    else fprintf(iobuf, "\n 0"); /* first column to file */
    /* loop for 15 more columns */
    for (i = 1; i < 16; i++) {
        htos(coladdr, str); /* convert column address */
        if (!*fname) puts(str); /* put to console */
        else fputs(str, iobuf); /* or to file */
        coladdr += colinc; /* increment the column address */
    }
}

/*
hex to string conversion
hex value is put into string with leading blanks
*/

```

(Continued on next page)

Program Monitor Package (Listing Continued, text begins on page 26)

Listing Three

```
htos(val, str)
{
    int val;
    char str[];

    int i;

    for (i=3; i>=0; i--) {          /* i is nibble number from the right */

        str[i] = (val >> ((3-i)*4)) 15;      /* isolate nibble */

        if (str[i] < 10)          /* for decimal values offset by '0' */
            str[i] += '0';
        else
            str[i] += '7';          /* '7' + 10 = 'A' */
    }
    str[4]='\\0';          /* terminate the string */

    for (i=0; str[i] == '0'; str[i++] = ' '); /* replace leading zeros */
    /* stopping at first digit */
}
```

End Listings

VANCE info systems is pleased to announce
THE MOST COMPLETE C FUNCTION LIBRARY AVAILABLE TO DATE!



"C.lib" is the most functional library available for software written in C, providing over 200 routines, extending the capabilities of C on the IBM PC. The library is available under the DeSmet (C Ware) C88 compiler, will be available in MicroSoft C, Lattice C and other C compilers, and runs using MS Dos 1.1 and later versions.

Routines are included to handle:

- System date & time I/O
- Date & time math functions
- Extended screen & keyboard I/O
- String & bit-field manipulation
- Degrees/radians conversions
- And contains dozens of functions compatible with Xenix and Unix
- String & numeric sorting
- Trigonometric & hyperbolic functions
- Asynchronous communications functions

"C.lib" features a unique windowing library transportable with Xenix function calls.

"C.lib" also includes functions to convert floating point numbers from MicroSoft to 8087 NDP format and vice versa.

Documentation is offered in an easy to use printed manual or on disk for your own printing needs, complete with programming examples and follow-up demo programs.

The "C.lib C FUNCTION LIBRARY is offered at only \$145, less than most available today.

For further information on "C.lib" please contact us at:

VANCE info systems

2818 clay street • san francisco, california 94115 • (415) 922-6539

IBM is a trademark of International Business Machines Corp.
C88 is a trademark of Computer Innovations, Inc.
Lattice is a trademark of Lattice Inc.
Xenix, MicroSoft C, and MS DOS are trademarks of Microsoft Inc.
Unix is a trademark of Bell Labs Inc.
C.lib is a trademark of VANCE info systems.

WRITE

The Writer's Really Incredible Text Editor lives up to its name! It's designed for creative and report writing and carefully protects your text. Includes many features missing from WordStar, such as sorted directory listings, fast scrolling, and trial printing to the screen. All editing commands are single-letter and easily changed. Detailed manual included. Dealer inquiries invited. WRITE is \$239.00.

BDS's C Compiler

This is the compiler you need for learning the C language and for writing utilities and programs of all sizes and complexities. We offer version 1.5a, which comes with a symbolic debugger and example programs. Our price is (postpaid) \$130.00.

Tandon Spare Parts Kits

One door latch included, only \$32.50.
With two door latches \$37.50.
Door latches sold separately for \$7.00.

All US orders are postpaid. We ship from stock on many formats, including: 8", Apple, Osborne, KayPro, Otrona, Epson, Morrow, Lobo, Zenith, Xerox. Please request our new catalog. We welcome COD orders.

Workman & Associates

112 Marion Avenue
Pasadena, CA 91106
(818) 796-4401



Circle no. 74 on reader service card.

Circle no. 79 on reader service card.

THE PROGRAMMER'S SHOP™

helps compare, evaluate, find products. Straight answers for serious programmers.

SERVICES

- Programmer's Referral List
- Compare Products
- Help find a Publisher
- Evaluation Literature free
- BULLETIN BOARD - 7 PM to 7 AM 617-826-4086
- Dealer's Inquire
- Newsletter
- Rush Order
- Over 300 products

"C" LANGUAGE

	LIST PRICE	OUR PRICE
MSDOS: C86-8087, reliable	\$395	call
Desmet with debugger	159	145
Instant C - Inter., fast, full	NA	500
Lattice 2.1 - improved	500	call
Microsoft C 2.x	500	349
Williams - NEW, debugger	500	call
CPM80 Ecosoft C-now solid, full	250	225
BDS C - solid value	150	125
MACINTOSH: Full, ASM	NA	385

Compare, evaluate, consider other Cs

BASIC

	ENVIRONMENT	LIST PRICE	OUR PRICE
BASCOM-86 - MicroSoft	8086	395	279
BASIC Dev't System	PCDOS	79	72
BASICA Compiler - BetterBASIC - 640K	PCDOS	NA	185
CB-86 - DRI	CPM86	600	439
Prof. BASIC Compiler	PCDOS	345	325
MACINTOSH COMPILER with BASICA syntax	MAC	NA	325

Ask about ISAM, other addons for BASIC

FEATURES

XSHELL Adds, Intelligent Batch files and other UNIX-like features to MSDOS. \$215.

PROLOG86 Interpreter for MSDOS includes tutorials, reference and good examples. Learn in first few hours. For Prototyping, Natural Language or AI. \$125.

PUBLIC DOMAIN Research - Free

6 months paid research gives you leverage, learning. We found, combined, added to the best. All run, have source in C or ASM. Order \$150 + get one free: Database, Editors, Modems, MSDOS RAMdisks & utils. Games in C.

EDITORS Programming

BRIEF - Intuitive, flexible	PCDOS	NA	195
C Screen with source	8080/86	NA	75
FINAL WORD - for manuals	8080/86	300	215
MINCE - like EMACS	CPM, PCDOS	175	149
PMATE - powerful	CPM	195	175
		8086	225
VEDIT - full, liked	CPM, PCDOS	150	119

UNIX PC

COHERENT - for "C" users	PClike	\$500	475
VENIX - "true V7" w/FTN	PClike	800	775
XENIX - "true S3" - rich	PC	1350	1295

Ask about run-times, applications, DOS compatibility, other alternatives. UNIX is a trademark of Bell Labs. Screen, Stat, Graphics.

LANGUAGE LIBRARIES

C Tools 1 - String, Screen	PCDOS	NA	115
C Tools 2 - OS Interface	MSDOS	NA	92
GRAPHICS: GraphiC-source	MSDOS	NA	195
HALO - fast, full	PCDOS	200	165
Greenleaf for C - full	MSDOS	NA	165
ISAM: C to dBASE-source	8086	150	140
BTRIEVE - many languages	MSDOS	245	215
Cindex + - no royalties	MSDOS	NA	395
PHACT - with C	MSDOS	NA	250
dBC - by Lattice	MSDOS	NA	250
PASCAL TOOLS - Blaise	PCDOS	NA	115
SCREEN: CView-source	PCDOS	NA	195
Databurst	MSDOS	225	215
PANEL-86-many languages	PCDOS	295	265
WINDOWS for C	PCDOS	NA	139

Ask about many others for FTN, BASIC, PASCAL, C-ISAM

RECENT DISCOVERIES

SCIL - Manage versions, changes to source code, documentation. Minimize confusion, disk space. Interactive. CPM 80, MSDOS \$ 349.

FORTRAN

	LIST PRICE	OUR PRICE
MS FORTRAN-86 - Impr.	MSDOS	\$350 \$ 225
Intel Fortran-86	IBM PC	NA 1400
DR Fortran-86 - full '77'	8086	500 349
PolyFORTRAN-XREF, Xtract	PCDOS	NA 165

OTHER PRODUCTS

Assembler & Tools - DRI	8086	200	159
CODESMITH-86 - debug	PCDOS	149	139
Disk Mechanic - rebuild	MSDOS	70	65
HS/FORTH	PCDOS	220	210
IQ LISP - full 1000K RAM	PCDOS	175	call
MBP Cobol-86 - fast	8086	750	695
MicroPROLOG	PCDOS	NA	285
Microshell improve CPM	8080	150	125
Microsoft MASM-86	MSDOS	100	85
MultiLink - Multitasking	PCDOS	295	265
PFIX-86 Debugger	MSDOS	195	169
PL/1-86	8086	750	495
PLINK-86 - overlays	8086	350	315
Polylibrarian - thorough	MSDOS	99	89
PolyMAKE	PCDOS	99	89
PROFILER-86 - easier	MSDOS	NA	125
PROFILER - flexible	MSDOS	NA	175
Programmers Tkt w/source	8086	NA	135
Prolog-86-Learn, Experiment	MSDOS	NA	125
READ CPM86 from PCDOS	PCDOS	NA	55
READ PCDOS on an IBM PC	CPM86	NA	55
TRACE86 debugger ASM	MSDOS	125	115
X Shell - IF-THEN-ELSE	MSDOS	225	215

Note: All prices subject to change without notice. Mention this ad. Some prices are specials.

Ask about COD and POs. All formats available.

0904

Call for a catalog, literature, and solid value

800-421-8006

THE PROGRAMMER'S SHOP™

128-D Rockland Street, Hanover, MA 02339.

Visa Mass: 800-442-8070 or 617-826-7531 MasterCard

Circle no. 54 on reader service card.

PROLOG-86™

Learn Fast, Experiment

1 or 2 pages of PROLOG would require 10 or 15 pages in "C."

Be familiar in **one evening**. In a **few days** enhance artificial intelligence programs included like:

- an Expert System
- Natural Language (generates dBASE display)

Intro Price: \$125 for PCDOS, CPM-86.

Full Refund if not satisfied.

CONTEST: "Artificial Intelligence Concepts"

\$1,000 Prize, Recognition for applications in PROLOG-86™ that teach, are clear, illustrate. Call for details. Deadline 11/31/84

SOLUTION SYSTEMS™

45-D Accord Park, Norwell, MA 02061

617-871-5435

Circle no. 80 on reader service card.

C Helper™

FIRST-AID FOR C PROGRAMS

Save time and frustration when analyzing and manipulating C programs. Use C HELPER's UNIX-like utilities which include:

- DIFF** and **CMP** - for "intelligent" file comparisons.
- XREF** - cross references variables by function and line.
- C Flow Chart** - shows what functions call each other.
- C Beautifier** - make source more regular and readable.
- GREP** - search for sophisticated patterns in text.

There are several other utilities that help with converting from one C compiler to another and with printing programs.

C Helper is written in portable C and includes both full source code and executable files for \$135 for MS-DOS, CPM-80 or CPM-86. Use VISA, Master Card or COD.

Call: 617-659-1571

Solution Systems™

335-D Washington Street
Norwell, MA 02061

Circle no. 81 on reader service card.

CP/M 2.2 Goes PUBLIC

by Bridger Mitchell and
Derek McKay

ZCPR, the widely used Z80 replacement for the standard Console Command Processor (CCP) supplied with CP/M 2.2, provides a much-appreciated search path feature. When you type a command (not prefixed by a drive designation), ZCPR first searches for the COM file on the logged-in drive in the current user number, next searches in the default user number (usually configured to user 0) on that drive, and finally looks in the default user number on drive A. The search terminates at the first matched filename; the command processor then loads that file and begins executing the program.¹

This popular feature relieves you of the necessity of keeping track of where you have stored many programs. By putting commonly used programs in the default user number directory, you can make COM files accessible from all directories. In Rick Conn's ZCPR2 and ZCPR3, you can also specify a particular search path for any specific file; this capability allows you to construct

grams—such as editors, formatters, data base managers, assemblers, and compilers—require overlays, work/swap files, libraries, dictionaries, and data base files that usually must reside in the directory of the logged-in drive and user number.

Currently three methods enable you to use such a program from more than one user number. You can use brute force to create copies of all of the files in several user numbers, a technique that very rapidly chews up disk storage. Or you can duplicate the directory entries (but not the files) into each user number with the public domain DUPUSR utility; this approach consumes more directory entries (but no additional disk space), and you run the risk of unintentionally deleting a key file should you forget to warm boot after erasing a duplicated directory entry. (After an erasure, the BDOS does not rebuild the disk group allocation vector and considers the “erased” sectors to be free for subsequent file-write operations.)

The third method is to load a search

“The PUBLIC file approach requires no additional disk space, directory entries or memory; it leaves the original file intact; and it protects files against accidental erasure.”

hierarchical directories.

The ZCPR search path is implemented at the CP/M command level and therefore works only for COM files. Unfortunately, this limits its usefulness for programs that refer to other “associated” (non-COM) files. Many major pro-

path utility, such as Michael Rubenstein's SETDRU, that intercepts BDOS disk function requests and redirects references to specified files. You first configure a filter utility with a list of associated files and their user numbers. The program (or a pre-program) then loads the filter into high memory where it traps and reinterprets requests for any associated file. SETDRU, however, uses some memory and cannot be run with other resident modules, such as EX and

*Bridger Mitchell & Derek McKay,
Plu*Perfect Systems, Box 1494,
Idyllwild, California 92349.*

XSUB (batch processing programs—the former public domain, the latter from Digital Research), The Background-er[®], and so on.

PUBLIC Files

Plu*Perfect Systems has developed a new "PUBLIC file" approach that is both simple and flexible. It requires no additional disk space, directory entries, or memory; it leaves the original files intact; and it protects files against accidental erasure. You simply set a single directory attribute bit—bit 2 of a filename—to make *any* file PUBLIC and accessible from all user numbers on that drive.² For example, once an editor and its swap and overlay files have been marked PUBLIC, you can edit a file stored in any user number by switching to that user number and running the editor. This requires only one copy and one directory entry of the editor and its associated files.

The key to the PUBLIC file approach is a patch that causes the BDOS to match a requested filename, regardless of user number, if that filename in the disk directory has the second attribute bit set. Because the directory carries the status of the file, the BDOS can determine, in a single scan of the directory, which file to open.

Our initial tests of this approach were encouraging but revealed one troublesome side effect: wild card (ambiguous) filename requests such as *??.COM* or **.OVL* frequently would match a PUBLIC file. This meant that DIR and most directory utilities would list all PUBLIC files in every user number. Much worse, *ERA** would unceremoniously wipe out every PUBLIC file!

After some experimentation, we modified the BDOS patch so that a wild card request will never match a PUBLIC filename. The only way to erase a PUBLIC file now is to type its exact (unambiguous) filename; you can do this from any user number. This also means that PUBLIC files are not normally displayed in directories, which gives them a special type of SYStem status. DIR will still display a PUBLIC file—if you give its exact name. But because you are more likely to use DIR to determine what files you have, whether you can remember their names or not, something more is needed.

Enter the PUBLIC program: a sup-



For the *first* time, a programmer's editor that is both intuitive *and* powerful ...and configurable to suit your style

The New Standard. No longer does an Editor have to be "in your way" to provide full power. By combining power *with* natural flow, the new advanced BRIEF is in a class by itself.

BRIEF lets you concentrate on programming. Your thoughts flow smoothly, intuitively. 15 minutes is all you need to become fully productive. You can then do precisely what you want quickly, with minimum effort and without dull repetitions.

BRIEF adapts to your style. You can use BRIEF without modification, because it's distributed with an "ideal" configuration. Or you can make any change you want, add any feature of your own. Reconfigure the whole keyboard or just the Function Keys. Change the way the commands work or just the start-up defaults.

Availability: PC DOS-compatible systems with at least 192K and one floppy drive are required. Though your initial copy is protected, an unprotected version is available when you register BRIEF.

Pricing: Only \$195... with discounts for volume end-users. A demonstration version is available for only \$10.

Win \$1,000 and substantial recognition for the Outstanding Practical BRIEF Macro. Other awards will also be given.

BRIEF'S PERFORMANCE IS NOT EQUALLED IN MICROS, MINIS AND MAINFRAMES

- | | |
|-------------------------------|--------------------------------|
| ■ Full UNDO (N Times) | ■ Windows (Tiled and "Pop Up") |
| ■ Edit Multiple Large Files | ■ Unlimited File Size |
| ■ True Automatic Indent for C | ■ Reconfigurable Keyboard |
| ■ Exit to DOS Inside BRIEF | ■ Online Help |
| ■ Uses All Available Memory | ■ Search for Complex Patterns |
| ■ Intuitive Commands | ■ Mnemonic Key Assignments |
| ■ Tutorial | ■ Horizontal Scrolling |
| ■ Repeat Keystroke Sequences | ■ Comprehensive Error Recovery |

PLUS a Complete, Powerful, Readable, Compiled MACRO Language

**Try BRIEF. Use the Demo...
or the full product for 30 days.**

**Call or write us...
617-659-1571**

**Solution
Systems™**

BRIEF is a trademark of Underware.
Solution Systems is a trademark of Solution Systems.

335-D Washington St., Norwell, MA 02061

Circle no. 67 on reader service card.

porting utility that (1) displays all PUBLIC files, (2) lets you make any file PUBLIC as long as it is the only file with that name in all user numbers, and (3) lets you return any PUBLIC file to the private sector.³ PUBLIC works because of a single exception to the "no wild card matching" rule: if the first (drive) byte of the file control block (fcb) is the CP/M wild card character "?", then the patched BDOS will match every directory entry, including PUBLIC files and erased files. (The source code for PUBLIC is in Listing Two, page 61.) Thus PUBLIC can search every directory entry on the disk to display all filenames of interest. We have also added this feature to the popular super directory (SD) utility; the extended version shows PUBLIC files in a separate listing category.

We considered, and rejected, the CP/M 3.0 and MP/M method of using the SYSTEM attribute bit to make a file PUBLIC if it is in user 0. In addition to requiring longer patches not easily shoe-horned into the BDOS, this approach is risky. Using STAT you can set the SYSTEM bit of a user 0 file, and if that file also exists in user *n*, the results can be disastrous: in servicing each extent of a file request logged into user *n*, the BDOS will open whichever entry first appears in the disk directory.

Patching the BDOS

We wrote the BDOS patch (Listing One, page 51) in Z80 code, which enabled us to fit it entirely within the BDOS. The functional changes are all in the internal BDOS function FINDNXT, which finds the next directory entry matching the supplied fcb. To gain space we recoded part of another routine GETNEXT and also used the free bytes at the end of the BDOS. Because this area occasionally is used for other patches—e.g., the Kelly Smith ARCHIVE utility—you should check your copy of the BDOS for possible conflicts. The flow of control is somewhat indirect; we solved the integer packing problem of fitting code fragments into rather tight, odd-length segments by trial and error! To install the patch on 8080 machines, you will need to rewrite the patch in 8080 opcodes and find some extra space in the BIOS.

One of the CDL MACRO Z80 assemblers or the Digital Research MAC is well suited for the patching task be-

cause it generates hex output for the exact code area without adding extra zero-filled bytes. After assembling the patch, use DDT to load it with an offset directly on top of the system image. If you don't have an assembler with this feature, you can assemble the patch, load it into an area above the SYSGEN image, and move the exact number of bytes into place with DDT.⁴

It goes without saying that you should thoroughly test any file system patching first on floppies and, of course, back up all files before beginning.

A CP/M 2.2 Bug

In working out the BDOS patch, we uncovered a CP/M bug: although the CP/M 2.2 Interface Guide states that the Rename and Erase functions return the values 0, 1, 2, or 3 on a successful operation, in fact CP/M 2.2 returns only 0 on success (and 0FFh on failure). Moreover, the Set Attribute function uses the same exit code, yet the Guide omits any mention of this function's return status value. It turns out that the correct code is actually shorter, and we have included it in our patch. It seems desirable to make this correction since a program that uses one of these functions may occasionally wish to access the BIOS directory buffer to retrieve the directory fcb for the matched entry.

Copying Files with Attribute Bits

Many programs have a preference for the private sector. In both the original CCP and the replacement ZCPR, the RENAME file command removes all attributes. Similarly, most file copy utilities (e.g., PIP, SWEEP, and DISK7) create the copy with no attribute bits. Thus, copying a PUBLIC file with PIP creates a private version; if the destination drive is the same as the source, this operation also erases the original PUBLIC source file. Many editors also do not preserve attribute bits, so editing a PUBLIC file turns it into a private citizen once again.

To support renaming and copying of PUBLIC files (as well as SYSTEM and Read/Only files), we have extended Frank Gaude's DISK7. The new version copies a file with its attribute bits intact and has an option for listing all PUBLIC files. In PUBLIC, SD, and

DISK7 we adopted the convention of displaying a filename character in lower case when its attribute bit is set. This convention readily identifies PUBLIC, SYSTEM, and Read/Only files. It is almost totally portable, and programmers revising other file copy, directory, and communications programs might consider adopting it.

PUBLIC files, combined with the ZCPR-type search path for loading programs, provide 90% of the control over user numbers that is needed on a CP/M system. What is still missing is the ability within most running programs to use private files in several user numbers. An excellent solution is to build file reference by user number into the language, as Leor Zollman has done in version 1.5 of his BDS-C. With this feature, for example, MINCE can concurrently edit files from several user numbers. Otherwise, you may be able to fix up major programs on a piecemeal basis. For example, we have patched the Perfect Writer version that was bundled with the Kaypro computers to add a "change user number" command.

The PUBLIC file approach does differ in some respects from a fully implemented search path because private and PUBLIC files with the same name cannot peacefully coexist on the same drive. Its principal advantages are that a single-pass search of the directory suffices for each open-file service request; the code occupies no extra space on a CP/M system and does not interfere with resident modules in high memory; and user programs needn't be modified.

We have found PUBLIC files to be a major convenience. For example, we used a PUBLIC editor and swap file in a single editing session to revise this article and the program files, which were stored in two separate user numbers, and to retrieve notes stored on a backup floppy from a third.

Availability

The PUBLIC file BDOS patches and PUBLIC utility program are copyrighted © 1984 by Plu*Perfect Systems, all rights reserved. We are releasing them for single-user use, with the restriction that they be used solely for noncommercial purposes. We hope that they will prove useful to many others. No

use of this code in commercial software or systems is permitted in any circumstances without advance written permission from Plu*Perfect Systems.

Plu*Perfect Systems markets an enhanced operating system for Kaypro computers that incorporates the PUBLIC file and command search path features and adds bidirectional typing capability. For users with other hardware, we can supply a disk for noncommercial use with all of the PUBLIC files (source code for the patches and the PUBLIC utility, copies of the extended SD.COM and DISK7.COM, an assembled hex-relocatable version of the patch, and a relocating loader) for \$12 on either an 8-inch SSD format disk or a number of 5¼-inch formats. Please specify the format you need.

Notes

1. Richard Conn, Ron Fowler, Keith Peterson, and Frank Wanchow developed ZCPR and placed it in the public domain. It is available from SIG/M User Group, Box 97, Iselin, NJ 08830. ZCPR3, written solely by Rick Conn, is available from Echelon, Inc., 101 First St., Los Altos, CA 94022. Ben Goldfarb has adapted portions of the Z80 ZCPR code to create an 8080 version of the CCP that incorporates the search path feature.

2. We do not use attribute bit 1 because we have already used it as a loading flag for keyboard definitions files for The Background—a resident background process utility currently available for Kaypro computers that also allows keys to be redefined dynamically.

3. PUBLIC is convenient but not essential; any program, such as Rick Conn's PROTECT, that can set and reset all attribute bits will move files to and from the public

sector. (Version 1.0 of PROTECT has one bug that causes it to fail to find large files on large hard disk systems because of an incorrectly calculated extent number.) Be careful not to set the PUBLIC attribute bit of a file that has the same name as a file in another user number.

4. Alternatively you could assemble the patch for a 32K CP/M system and install it in the MOVCPM image. However, this would require generating and installing a new bitmap for the modified bytes. We find it easier to have separate system images for the handful of different system sizes needed for any one machine.

DDJ

Reader Ballot

Vote for your favorite feature/article.
Circle Reader Service No. 194.

CP/M 2.2 (Text begins on page 48)

Listing One

```
2 .settim 21:30:00
3 .setdat 04/13/84
6 .remark ~
7
8          -- PUBPATCH --
9 A CP/M 2.2 BDOS modification to support the PUBLIC filetype.
10
11 -----
12 Copyright (c) 1984 -- All rights reserved.
13
14          Plu*Perfect Systems
15          P. O. Box 1494
16          Idyllwild CA 92349
17 -----
18
19 Attribute bit 2 of a filename signifies a PUBLIC file,
20 accessible by its unambiguous filename from all user
21 numbers.
22
23 PUBLIC files are not accessible via the usual ambiguous
24 filenames (e.g. *.* or ABC.D?F), to prevent unintentional
25 erasure and avoid directory clutter.
26
27 Directory entries for PUBLIC files are, however, accessible
28 via ambiguous filenames by using the BDOS search-for-first,
29 search-for-next functions with a '?' in the drive-byte of the
30 fcb. Extended versions of SD and DISK7 displays PUBLIC files.
31
32 To erase a PUBLIC file, use "ERA unambiguous-filename". Or use
33 DISK7. Or change it to a private file and then erase with a
34 wildcard erase command.
35
36 The PUBLIC.COM utility is available to make files either
```

(Continued on next page)

Listing One

```
37 Public or private and to list the current Public files.
38
39 If another utility is used to set the Public attribute bit,
40 avoid creating multiple files with the same name on the same
41 drive, unless all of them are private. (PUBLIC.COM checks
42 for this situation and prevents a conflict.)
43
44 The RENAME command removes all attributes, so RENaming a
45 Public file will make it private, R/W, DIR in its original
46 user number.
47 ~

50 .remark ~
51          --- TO INSTALL ---
52
53 1a. Determine the BIOSBASE address of your system in memory
54     by subtracting 3 from the warm-boot address in memory:
55         DDT
56         L0
57         subtract 3 ==> BIOSBASE address
58 1b. Subtract 1600H to determine the CCPBASE_MEMORY address.
59 1c. Assemble PUBPATCH for these addresses.
60
61     Either: use CDL's MACROIII assembler:
62
63         MACROIII PUBPATCH A:DHK
64
65     Or: convert the pseudo-opcodes to your assembler's
66         pseudo-ops and assemble into a HEX file.
67         e.g.      .loc      ==>      org
68                  =          ==>      set
69                  =\         ==>      ????, etc.
70
71 2. Create a system image for the SYSGEN operation.
72     There are two ways to get the image:
73
74     a. Either use SYSGEN to extract a system image from a disk
75         in the usual manner --
76         SYSGEN
77         source drive? A
78         destination drive? <CR>
79         SAVE pp ORIG.SYS.      Use pp=50 pages or so to get
80                                the entire BIOS.
81
82     b. Or generate a new system --
83         MOVCPM ss *             where ss=64 for a 64K system,
84         SAVE pp ORIG.SYS       or whatever you are running.
85
86 3. Find the base address of the Command Processor in the image
87     DDT ORIG.SYS
88     Look for the command processor at 980H:
89     You recognize it by two JMP instructions, followed by
90     the command buffer (containing a Digital Research
91     copyright notice, in the case of the original CCP):
92     L980
```



```

93         D980
94         Call that address CCPBASE_IMAGE (normally 9800H).
95         (If you don't find it, you have a non-standard system, and
96         your user's manual should have a memory map. See, e.g.,
97         the Compupro Disk 1 Controller manual, sec. 6.4).
98
99     4. Calculate the offset needed to cause the PUBPATCH.HEX file to
100        load on top of the BDOS image. 'offset' will satisfy:
101
102        CCPBASE_IMAGE = CCPBASE_MEMORY + offset
103
104     5. Create a new system image containing the patch:
105
106        DDT ORIG.SYS
107        IPUBPATCH.HEX
108        Roffset
109        G0
110        SAVE pp NEWSYS.SYS
111
112     6. Finally, put the new system on a FLOPPY disk for testing:
113
114        SYSGEN NEWSYS.SYS
115        <CR>
116        destination_drive
117
118        -----
119        Code also corrects a CP/M 2.2 bug that caused
120        Rename, Set Attribute, and Delete File functions
121        to return 0 status on success instead of 0,1,2,3
122        per CP/M 2.2 Installation Guide.
123        -----
124
125    ~
126
127    .phex
EA00 129 biosbase = \ "Enter BIOS base address for PUBPATCH (xxxxH) "
130
DC00 131 bdosbase = biosbase - 0e00h
D400 132 ccpbase = biosbase - 1600h
133
134    .remark ~
135    biosbase = 0ea00h ; kaypro10 v 1.9
136              = 0fa00h ; kaypro 2
137              = 0dc00h ; h89
138    ~
139    ;      Internal BDOS locations:
140    ;
E32D 141 FINDNXT =      bdosbase+072Dh
E205 142 NXENTRY =      bdosbase+0605h
E1F5 143 CKFILPOS =      bdosbase+05F5h
E17F 144 MOREFLS =      bdosbase+057Fh
E15E 145 FCB2HL =      bdosbase+055Eh
E307 146 SAMEXT =      bdosbase+0707h
E1FE 147 STFILPOS =      bdosbase+05FEh
DF01 148 SETSTAT =      bdosbase+0301h
E9D9 149 SAVEFCB =      bdosbase+0DD9h
E9D8 150 COUNTER =      bdosbase+0DD8h
E9EA 151 FILEPOS =      bdosbase+0DEAh
DF45 152 STATUS =      bdosbase+0345h
E9D4 153 FNDSTAT =      bdosbase+0DD4h

```

(Continued on next page)

Listing One

```

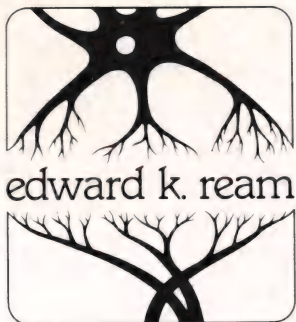
E154      154  CHKWPRT =      bdosbase+0554h
E144      155  CHKROFL =      bdosbase+0544h
E318      156  FINDFST =      bdosbase+0718h
E26B      157  SETFILE =      bdosbase+066bh
E1C6      158  DIRWRITE =      bdosbase+05c6h
E1F5      159  CKFILPOS =      bdosbase+05f5h
E8D7      160  DELFILE =      bdosbase+0cd7h
E9C5      161  EXTMASK =      bdosbase+0dc5h
E9D2      162  CLOSEFLG =      bdosbase+0dd2h
E9D3      163  RDWRTFLG =      bdosbase+0dd3h
E524      164  GETEMPTY =      bdosbase+0924h
E45A      165  OPENIT1 =      bdosbase+085ah
E0BB      166  STRDATA =      bdosbase+04bbh
DF01      167  SETSTAT =      bdosbase+0301h
DF05      168  IOERR1 =      bdosbase+0305h
E178      169  SETS2B7 =      bdosbase+0578h
          170  ;

          173      .pabs
E32D      174      .loc      FINDNXT
          175
E32D      21  0000      176  fnxt0:  lxi      h,0
E330      22  E399      177      shld     pflag      ;initialize PUBLIC & wildcard flags
E333      4C          178      mov      c,h          ;0
E334      CD E205      179      call     NXENTRY
E337      CD E1F5      180      call     CKFILPOS
E33A      2866         181      jrz      nomatch          ;if done
E33C      2A E9D9      182      lhld     SAVEFCB
E33F      EB          183      xchg
E340      1A          184      ldax     d          ;de=user-fcb
E341      FEE5         185      cpi      0E5h      ;if Getempty fn wants first
E343      2807         186      jrz      fnxt1      ;..deleted file slot in directory
E345      D5          187      push     d
E346      CD E17F      188      call     MOREFLS
E349      D1          189      pop      d
E34A      3056         190      jrnc     nomatch          ;if no more files
          191      ;
E34C      CD E15E      192  fnxt1:  call     FCB2HL          ;hl=directory fcb
E34F      3A E9D8      193      lda      COUNTER
E352      47          194      mov      b,a          ;b=count
E353      0E00         195      mvi      c,0          ;c=byte #
E355      B7          196      ora      a          ;COUNTER=0 ==> Search fn
E356      2847         197      jrz      matched        ;..so match every entry
          198      ;
E358      79          199  fnxt2:  mov      a,c          ;get byte #
E359      FE0D         200      cpi      13
E35B      2836         201      jrz      nxtbyte        ;omit S1 byte
E35D      1A          202      ldax     d          ;get user-fcb char
E35E      FE3F         203      cpi      '?'
E360      2005         204      jrnz     fnxt3
E362      32 E39A      205      sta      qflag      ;flag wildcard
E365      182C         206      jr       nxtbyt

```

(Continued on page 56)

RED



FULL SCREEN EDITOR with FULL SOURCE CODE in C for CP/M 68K or CP/M 80

- RED is a powerful yet simple full screen text editor for both programmers and writers.
- RED features 17 commands including block move, block copy, search and substitute.
- RED comes with *full* source code in standard C. RED works as is with the BDS C, Aztec CII and Digital Research Compilers.
- RED supports *all* features of your terminal. You tailor RED to your terminal with an easy-to-use configuration program.
- RED handles files as large as your disk.
- RED is guaranteed. If for any reason you are not satisfied with RED, your money will be refunded promptly.

Price: \$95.

Call today for more valuable information:
(608) 231-2952

To order, send a check or money order to:

**Edward K. Ream
1850 Summit Avenue
Madison, Wisconsin 53705**

Your order will be mailed to you within one week. Sorry, I can not handle credit cards. Please do not send purchase orders unless a check is included. RED is distributed only on 8 inch CP/M format disks, but other disk formats are available through third parties.

Dealer inquiries invited.

At Last! bds C... Ver. 1.5

Including a new dynamic debugger Still the choice of professionals

- Compiler option to generate special symbol table for new dynamic debugger by David Kirkland. (With the debugger, the distribution package now requires two disks.)
- Takes full advantage of CP/M* 2.x, including random-record read, seek relative to file end, user number prefixes, and better error reporting.
- Clink option to suppress warm-boot
- New library file search capabilities
- New, fully-indexed 180 page manual
- * CP/M is a trademark of Digital Research, Inc.

V 1.5 \$120.00

V 1.46 \$115.00
(needs only 1.4 CP/M)

Other C compilers and
C related products
available . . . Call!

TERMS: CHECK,
MONEY ORDER, C.O.D.,
CHARGE CARD
HOURS: 9 am—5 pm
Monday—Friday
(316) 431-0018

IT'S HERE!

MONEY MATH

- Uses BCD internal representation.
- You choose from two types of rounding.
- Configurable exception handling
- Distributed with 12 digits precision. Easily configured for more or less
- Excess 64 exponents

SOURCE INCLUDED **\$5000**



include \$2.50 for postage and handling

Circle no. 22 on reader service card.

A Professional Quality Z80/8080 Disassembler

REVAS Version 3

Uses either ZILOG or 8080 mnemonics
Includes UNDOCUMENTED Z80 opcodes
Handles both BYTE (DB) & WORD (DW) data
Disassembles object code up to 64k long!
Lets you insert COMMENTS in the disassembly!

A powerful command set gives you:

INTERACTIVE disassembly
Command Strings & Macros
On-line Help
Calculations in ANY Number Base!
Flexible file and I/O control
All the functions of REVAS V2.5

REVAS:

Is fully supported with low cost user updates
Runs in a Z80 CPU under CP/M*
Is normally supplied on SSSD 8" diskette
Revas V 3...\$90.00 Manual only...\$15.00
California Residents add 6½% sales tax

REVASCO

**6032 Charlton Ave., Los Angeles, CA. 90056
(213) 649-3575**

*CP/M is a Trademark of Digital Research, Inc.

Circle no. 61 on reader service card.

Listing One

```

E367 7E          207 ;
E368 FEE5       208 fnxt3: mov    a,m      ;get directory-fcb char
E36A 79         209          cpi    0E5h    ;check for blank/deleted file
E36B 2814       210          mov    a,c      ;A = byte #
E36D B7         211          jrz    chkext  ;if a deleted file, omit user # check
E36E 2011       212          ora    a
E370 23         213          jrnz   chkext  ;or if not user # byte
E371 23         214          inx    h      ;else check for PUBLIC file
E372 CB7E       215          inx    h
E374 2B         216          bit    7,m      ;..at attribute bit 2
E375 2B         217          dcx    h
E376 2809       218          dcx    h
                219          jrz    chkext  ;if not PUBLIC, match on user #
                220 ;

                221 ; the file is PUBLIC
                222 ; -- but is BDOS looking for an empty directory slot?
E378 78         223          mov    a,b      ;if COUNTER=1, this is a Getempty request

E379 3D         224          dcr    a      ;
E37A 28B1       225          jrz    fnxt0    ;..so go to next file
E37C 32 E399    226          sta    pflag  ;else flag the file PUBLIC,
E37F 1812       227          jr     nxtbyt  ;..and omit matching user #
                228 ;
E381 FE0C       229 chkext: cpi    12      ; (A=byte #)
E383 1A         230          ldax   d
E384 2805       231          jrz    tstext  ;extent byte(#12) is special case
E386 96         232          sub    m      ;compare the characters
E387 E67F       233          ani    07fh  ;..excluding attribute bits
E389 1806       234          jr     extdone
E38B C5         235 tstext: push   b      ;check for same extent
E38C 4E         236          mov    c,m
E38D CD E307    237          call   SAMEXT
E390 C1         238          pop    b
E391 209A       239 extdone:jrnz   fnxt0    ;if mismatch, get next file
                240 ;
E393 13         241 nxtbyt: inx    d      ;chars match, bump to next byte
E394 23         242          inx    h
E395 0C         243          inr    c      ;byte # ++
E396 10C0       244          djnz   fnxt2    ;count--
                245 ;
                246 ; here if-- COUNTER > 1 and filenames match
                247 ;
                248 ;Test for PUBLIC file and wild-card combination:
E399           249          ;
E39A           250 pflag = .+1      ;flags initially = 0, but
                251 qflag = .+2      ; = COUNTER-1 if PUBLIC
                252          lxi    h,.-.      ; = '?' (3Fh) if '?' in fcb+1...
E398 21 0000    253          mov    a,l      ;if file is PUBLIC
E39B 7D         254          ana    h      ;..and there's a wildcard
E39C A4         255          ;(3Fh & 1...n) ==> NZ
E39D 208E       256          jrnz   fnxt0    ;..get next directory entry
                257 ;
                258 ; here if--

```



```

259 ; (a) non-PUBLIC filenames match,
260 ; or (b) find-all-files (searchfirst/searchnext functions
261 ; with drive byte = '?')
262 ; or (c) delete unambiguous PUBLIC-filename.
263 ;
E39F C3 E5AE 264 matched: jmp PATCH1
265 ;
E3A2 CD E1FE 266 nomatch: call STFILPOS
E3A5 7D 267 mov a, l ; l=0ffh
E3A6 C3 DF01 268 jmp SETSTAT
269 ;

272 ; the ERASE FILE routine -- in a new location
273 ;
274 ; Routine is split, with remainder stuffed
275 ; into free bytes at end of BDOS.
276 ;
E3A9 CD E154 277 ERAFILE: call CHKWPRT ; write-protect aborts
E3AC 0E0C 278 mvi c, l2
E3AE CD E318 279 call FINDFST
E3B1 CD E1F5 280 erafl: call CKFILPOS ; check for 'E5' case
E3B4 C8 281 rz
E3B5 CD E144 282 call CHKROFL ; read-only file aborts
E3B8 CD E15E 283 call FCB2HL
E3BB C3 E9EE 284 jmp PATCH2

```

(Continued on next page)

The BDS C Compiler . . . "Performance: Excellent. Documentation: Excellent. Ease of Use: Excellent."

That's what *InfoWorld* said when we introduced the BDS C Compiler four years ago. Today, the updated **BDS Version 1.5** is even *better*.

First, the BDS is still the *fastest* CPM/80-C compiler available anywhere.

Next, the new revised user's guide comes complete with tutorials, hints, error messages and an easy-to-use index — the perfect manual for beginner or seasoned pro.

Plus, the following, all for *one price*: Upgraded file searching ability for all compiler/linkage system files. Enhanced file I/O mechanism that lets you manipulate files anywhere in your system. Support system for *float* and *long* via library functions. An interactive symbolic debugger. Dynamic overlays. Full source code for libraries and run-time package. Sample programs include utilities and games.

Don't waste another minute on a slow language processor. Order now.

Complete Package (two 8" SSDD disks, 181-page manual): **\$150**. Free shipping on prepaid orders inside USA. VISA/MC, C.O.D.'s, rush orders accepted. Call for information on other disk formats.

BDS C is designed for use with CPM-80 operating systems, version 2.2 or higher. It is not currently available for CPM-86 or MS-DOS.

BD Software

BD Software, Inc.
P.O. Box 2368
Cambridge, MA 02238
(617) 576-3828

Announcing a

TOTAL PARSER GENERATOR

<GOAL> :: = <RAPID> <COMPILER> <DESIGN>

SLICE YOUR COMPILER DEVELOPMENT TIME

An LR(1) parser generator and several sample compilers, all in Pascal for your microcomputer.

- Generates parser, lexical analyzer and skeleton semantics
- Universal, state-of-the art error recovery system
- Adaptable to other languages
- Interactive debugging support
- Thorough documentation
- **TURBO PASCAL™ INCLUDED FREE OF CHARGE**
- Includes mini-Pascal compiler, assembler, simulator in SOURCE

SPECIAL INTRODUCTORY OFFER \$1995

OPARSER™ runs on IBM PC/DOS in Turbo Pascal. Parser generator in object form; all else in source. OPARSER takes a grammar and generates a correct, complete, high-performance compiler with skeleton semantics in Pascal source. Easy to add full semantics for YOUR application. Excellent for industrial and academic use. An accompanying textbook (SRA publishers) available in 1985. Training can be arranged.

Educational and quantity discounts available. Check, money order, Mastercard, Visa. California residents add 6.5% sales tax.

WRITE OR CALL FOR FREE BROCHURE.

Technical details: call 408/255-5574. Immediate delivery. CALL TODAY!

QCAD

SYSTEMS, INC.

1164 Hyde Ave., San Jose, CA 95129

TOLL FREE: 800-538-9787

(California residents call 408/255-5574)

™ Turbo Pascal is a registered trademark of Borland International.

Circle no. 5 on reader service card.

Circle no. 56 on reader service card.

Listing One

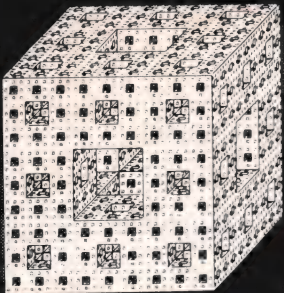
```

E3BE          285 LAST1 = .          ;must be <= bdosbase+07BEh
              286 ;
              287 ;
              288
              289
              290 ;Remainder of ERASE routine goes at end of BDOS
              291 ;
E9EE          292          .loc      bdosbase+0deeh ;there are 18 spare bytes
              293
E9EE          294 PATCH2:
E9EE 36E5      295 eraf2: mvi        m,0E5h ;install erase mark
E9F0 0E00      296          mvi        c,0
E9F2 CD E26B   297          call      SETFILE ;clear file's space in bitmap
E9F5 CD E1C6   298          call      DIRWRITE ;write directory sector
E9F8 CD E32D   299 eraf3: call      FINDNXT ;look for next entry
E9FB C3 E3B1   300          jmp        eraf1
E9FE          301 LAST2 = .          ;must be <= bdosbase+0E00h
              302 ;

              305 ;rewrite last part of bdos GETNEXT routine to gain space
              306 ;
E571          307          .LOC      bdosbase+0971h
              308 ;
E571 2815      309 gnxt0: jrz        gnxt1  ;( overlaying jz gtnext1)
E573 47        310          mov        b,a      ;extent byte
E574 3A E9C5   311          lda        EXTMASK
E577 A0        312          ana        b
E578 21 E9D2   313          lxi        h,CLOSEFLG
E57B A6        314          ana        m
E57C 2812      315          jrz        gnxt2  ;must read next extent
E57E CD E45A   316 gnxt3: call      OPENIT1 ;open current extent
E581 CD E0BB   317 gnxt4: call      STRDATA ;update rec#, extent#,...
E584 AF        318          xra        a
E585 C3 DF01   319 jsetst: jmp      SETSTAT
              320 ;
              321 ;      have overflowed normal extent, check s2 byte
E588 23        322 gnxt1: inx        h      ;shorter code, replacing
E589 23        323          inx        h      ;lxi b,2 & dad b
E58A 34        324          inr        m      ;bump s2 byte
E58B 7E        325          mov        a,m
E58C E60F      326          ani        0fh
E58E 2818      327          jrz        gnxt5  ;error if too many extents
E590 0E0F      328 gnxt2: mvi        c,15  ;open the next extent
E592 CD E318   329          call      FINDFST
E595 CD E1F5   330          call      CKFILPOS
E598 20E4      331          jrnz       gnxt3
E59A 3A E9D3   332          lda        RDWRFLG ;no extant extent
E59D 3C        333          inr        a      ;..if reading, can't open one
E59E 2808      334          jrz        gnxt5
E5A0 CD E524   335          call      GETEMPTY ;writing, so get next free entry
E5A3 CD E1F5   336          call      CKFILPOS
E5A6 20D9      337          jrnz       gnxt4  ;and if no error, save the data
E5A8 CD DF05   338 gnxt5: call      IOERR1 ;set error &

```

(Continued on page 60)



WALTZ LISP^(TM)

The one and only **adult** Lisp system for CP/M users.

Waltz Lisp is a very powerful and complete implementation of the Lisp programming language. It includes features previously available only in large Lisp systems. In fact, Waltz is substantially compatible with Franz (the Lisp running under Unix), and is similar to MacLisp. Waltz is perfect for Artificial Intelligence programming. It is also most suitable for general applications.

Much faster than other microcomputer Lisps. • Long integers (up to 611 digits). Selectable radix • True dynamic character strings. Full string operations including fast matching/extraction. • Flexibly implemented random file access. • Binary files. • Standard CP/M devices. • Access to disk directories. • Functions of type lambda (expr), nlambda (fexpr), lexpr, macro. • Splicing and non-splicing character macros. • User control over all aspects of the interpreter. • Built-in prettyprinting and formatting facilities. • Complete set of error handling and debugging functions including user programmable processing of undefined function references. • Virtual function definitions. • Optional automatic loading of initialization file. • Powerful CP/M command line parsing. • Fast sorting/merging using user defined comparison predicates. • Full suite of mapping functions, iterators, etc. • Assembly language interface. • Over 250 functions in total. • The best documentation ever produced for a micro Lisp (300+ full size pages, hundreds of illustrative examples).

Waltz Lisp requires CP/M 2.2, Z80 and 48K RAM (more recommended). All common 5" and 8" disk formats available.

PROCODE^(TM)
INTERNATIONAL

15930 SW Colony Pl.
Portland, OR 97224

Unix* Bell Laboratories.
CP/M* Digital Research Corp.

Version 4.4

(Now includes Tiny Prolog
written in Waltz Lisp.)

\$169*

*Manual only: \$30 (refundable with order). All foreign orders: add \$5 for surface mail, \$20 for airmail. COD add \$3. Apple CP/M and hard sector formats add \$15.

Call free **1-800-LIP-4000** Dept. #11
In Oregon and outside USA call 1-503-684-3000

Circle no. 52 on reader service card.

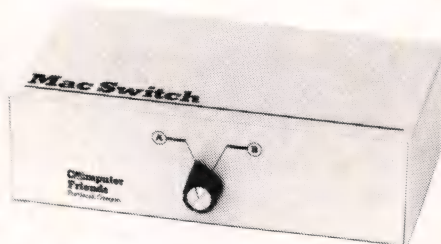
Re-ink any fabric ribbon for less than 5¢. Extremely simple operation. We have a **MAC INKER** for any printer. Lubricant ink safe for dot matrix printheads. Multi-colored inks, uninked cartridges available. Ask for brochure. Thousands of satisfied customers.

\$54⁹⁵ +



Mac Switch lets you share your computer with any two peripherals (serial or parallel). Ideal for word processors—never type an address twice. Ask us for brochure with tips on how to share two peripherals with **MAC SWITCH**. Total satisfaction or full refund.

\$99⁰⁰



Order Toll Free 1-800-547-3303

**Mac Inker
& MacSwitch**

**Computer
Friends**

6415 SW Canyon Court
Suite #10
Portland, Oregon 97225
(503) 297-2321

Circle no. 11 on reader service card.

NEW FEATURES

(Free update for our early customers!)

- Edit & Load multiple memory resident files.
- Complete 8087 assembler mnemonics.
- High level 8087 support. Full range transcendentals (tan, sin, cos, arctan, logs and exponentials) Data type conversion and I/O formatting.
- High level interrupt support. Execute Forth words from within machine code primitives.
- 80186 Assembler extensions for Tandy 2000, etc.
- Video/Graphics interface for Data General Desktop Model 10

HS / FORTH

- Fully Optimized & Tested for:
IBM-PC IBM-XT IBM-JR
COMPAQ EAGLE-PC-2
TANDY 2000 CORONA
LEADING EDGE

(Identical version runs on almost all MSDOS compatibles!)

- Graphics & Text (including windowed scrolling)
- Music - foreground and background includes multi-tasking example
- Includes Forth-79 and Forth-83
- File and/or Screen interfaces
- Segment Management Support
- Full megabyte - programs or data
- Complete Assembler (interactive, easy to use & learn)
- Compare
BYTE Sieve Benchmark jan 83
HS/FORTH 47 sec BASIC 2000 sec
w/AUTO-OPT 9 sec Assembler 5 sec
other Forths (mostly 64k) 70-140 sec
FASTEST FORTH SYSTEM AVAILABLE.

**TWICE AS FAST AS OTHER
FULL MEGABYTE FORTHS!**

(TEN TIMES FASTER WHEN USING AUTO-OPT!)

HS/FORTH, complete system only: \$250.

VISA Visa Mastercard
Add \$10. shipping and handling

**HARVARD
SOFTWARES**

PO BOX 339
HARVARD, MA 01451
(617) 456-3021

Circle no. 33 on reader service card.

Listing One

```

E5AB C3 E178      339          jmp      SETS2B7 ;..don't close file
                  340      ;
                  341      ;
                  342      ;          use space (14 bytes) for fragment from FINDNXT routine:
                  343      ;
E5AE 3A E9EA      344 PATCH1: lda      FILEPOS
E5B1 E603          345          ani      03h
E5B3 32 DF45      346          sta      STATUS ;save its directory buffer index
E5B6 21 E9D4      347          lxi      h,FNDSTAT
                  348      ;
                  349          .remark ~
                  350 The original CP/M 2.2 code removed below is erroneous, and
                  351 causes BDOS Erasefile, Renamefile, Setattribute functions
                  352 to return A=0 on success rather than the directory index
                  353 (0,1,2 or 3) specified in the Interface Guide.
                  354
                  355 ;;      mov      a,m
                  356 ;;      ral
                  357 ;;      rnc
                  358 ;;      xra      a
                  359 ~
                  360 ;
E5B9 77           361          mov      m,a      ;also save it for use
E5BA C9           362          ret          ;..by Erase,Rename,Set Attribute fns
                  363 ;
E5BB             364 LAST3 = .      ;must be <= bdosbase+09BCh
                  365 ;
                  366 ;
                  367 ;patch ERAFILE reference to its new location
                  368 ;
E8DA             369          .loc      (DELFILE+3)
E8DA CD E3A9      370          CALL     ERAFILE
                  371 ;
                  372          .end

```

+++++ Symbol Table +++++

BDOSBA DC00	BIOSBA EA00	CCPBAS D400	CHKEXT E381
CHKROF E144	CHKWPR E154	CKFILP E1F5	CLOSEF E9D2
COUNTE E9D8	DELFIL E8D7	DIRWRI E1C6	ERAF1 E3B1
ERAF2 E9EE	ERAF3 E9F8	ERAFIL E3A9	EXTDON E391
EXTMAS E9C5	FCB2HL E15E	FILEPO E9EA	FINDFS E318
FINDNX E32D	FNDSTA E9D4	FNXT0 E32D	FNXT1 E34C
FNXT2 E358	FNXT3 E367	GETEMP E524	GNXT0 E571
GNXT1 E588	GNXT2 E590	GNXT3 E57E	GNXT4 E581
GNXT5 E5A8	IOERR1 DF05	JSETST E585	LAST1 E3BE
LAST2 E9FE	LAST3 E5BB	MATCHE E39F	MOREFL E17F
NOMATC E3A2	NXENTR E205	NXTBYT E393	OPENIT E45A
PATCH1 E5AE	PATCH2 E9EE	PFLAG E399	QFLAG E39A
RDWRTF E9D3	SAMEXT E307	SAVEFC E9D9	SETFIL E26B
SETS2B E178	SETSTA DF01	STATUS DF45	STFILP E1FE
STRDAT E0BB	TSTEXT E38B		

Listing Two

CP/M RMAC ASSEM 1.1

```

                                title 'PUBLIC.ASM 4-12-84 (c) 1984 Plu*Perfect Systems'
                                PAGE 55

                                ;
000A =    vers      equ      1$0
                                ;
                                ;Utility to set/clear attribute bit 2  of filename.
                                ;For use with Plu*Perfect Systems' Public-file BDOS patch.
                                ;
                                ;usage: PUBLIC [d:]                -- lists PUBLIC files (on d:)
                                ;      PUBLIC [d:]filename.typ    -- makes filename.typ PUBLIC
                                ;      PUBLIC [d:]filename.typ [X] -- makes filename.typ private
                                ;      filename,type must be unambiguous - no wildcards
                                ;
                                ;prints drive/user number of all files of specified name,
                                ;  with attribute bits displayed as lower-case characters
                                ;
                                ;if just one such file found:
                                ;      if X, clears attribute bit - making file private in orig user #
                                ;      else,  sets attribute bit - making file PUBLIC
                                ;
                                ;
                                maclib z80

0000 =    FALSE    EQU      0
FFFF =    TRUE     EQU      NOT FALSE

                                ;
0000 =    LISTPUBBIT      equ      0
0001 =    MAKPUBBIT      equ      1
007C =    fence         equ      '|'

                                ;
0000 =    NULL      equ      0
0007 =    BELL      equ      'G'-'@'
000D =    CR        equ      0dh
000A =    LF        equ      0ah
001B =    ESC       equ      1bh
0020 =    SPACE     EQU      ' '
007F =    DEL       EQU      7FH

000E =    drivefn equ      14
000F =    openfn  equ      15
0010 =    closefn equ      16
0011 =    srchfstfn equ     17
0012 =    srchnxtfn equ     18
0013 =    deletefn equ     19
0014 =    readfn  equ      20
0015 =    writefn equ      21
0016 =    makefn  equ      22
0017 =    renamefn equ     23
0018 =    logvecfn equ     24
0019 =    curdskfn equ     25
001A =    dmafn   equ      26
001E =    setattrfn equ     30
001F =    getaddrfn equ     31

```

(Continued on next page)

CP/M 2.2 (Listing Continued, text begins on page 48)

Listing Two

```

0020 =      userfn equ      32
0021 =      readrfn equ    33
0022 =      writerfn equ   34
0023 =      sizefn equ     35
;
000F =      reccnt equ     15
0020 =      currec equ     32
0021 =      r0 equ        33
0080 =      tbuff equ     80h
005C =      fcb equ       5ch
006C =      fcb2 equ      fcb+16
;
0000 =      BOOT equ      0000h
0005 =      BDOS equ      0005h
FFFF =      ADDRESS equ   0FFFFh ;filled in by loader
0E00 =      bdoslen equ    0E00h

dobdos macro num,arg
    if not nul arg
        lxi de,arg
    endif
    mvi c,num
    call 5
endm

;
print macro msg
    if not nul msg
        lxi d,msg
    endif
    call printde
endm

;
$-MACRO
PAGE
aseg
org 100h

0100
;
0100 C38702 top: jmp start
;
0103 0D0A0A5542banner: db CR,LF,'PUBLIC v '
010E 312E30 db vers/10+'0','.',(vers mod 10)+'0'
0111 2028632920 db ' (c) 1984 Plu*Perfect Systems$'
;
012F 0D0A0A5573usage: db CR,LF,LF,'Usage: '
013A 5055424C49 db 'PUBLIC [d:] -- list PUBLIC files (on d:)'
016C 0D0A202020 db CR,LF,' '
0176 5055424C49 db 'PUBLIC [d:]file.ext -- make file PUBLIC'
019F 0D0A202020 db CR,LF,' '
01A9 5055424C49 db 'PUBLIC [d:]file.ext X -- make file private'
01D3 0D0A0A24 db CR,LF,LF,'$'
;
01D7 0D0A0A5468header: db CR,LF,LF,'The currently PUBLIC files are:',CR,LF,'$'
;

```



```

01FC 2073657420tomsg: db      ' set to ==> $'
0209 5055424C49pubnam: db      'PUBLIC  $'
0213 5052495641privnam:db      'PRIVATE  $'
021D 20203D3D3Eismsg: db      ' ==> is already $'

;
022F 0D0A072A2Amultimsg:db      CR,LF,BELL,'*** Multiple copies, can''t change!$'
0255 0D0A072A2Anonemsg:db      CR,LF,BELL,'*** No file!$'
0265 0D0A072A2Acantmsg:db      CR,LF,BELL,'*** Can''t make file $'
027D 0D0A284E6Fnopubs: db      CR,LF,'(None.)$'

;
;
0287      start:  sspd      ustack
028B 316505      lxi      sp,stack
028E      print      banner
0294 215D00      lxi      h,fcbl+1
0297 7E          mov      a,m
0298 FE20        cpi      SPACE
029A 210D05      lxi      h,flags
029D      res      LISTPUBBIT,m
029F      jrnz      setup
02A1      setb      LISTPUBBIT,m
02A3      print      header

;
02A9      setup:  dobdos   dmafn,buf      ;set directory buffer
02B1      dobdos   curdskfn      ;save user's drive
02B6 320F05      sta      drive
02B9 321105      sta      udrive
02BC 3A5C00      lda      fcb      ;check for specified drive
02BF B7          ora      a
02C0      jrz      savusr
02C2 3D          dcr      a
02C3 320F05      sta      drive      ;login specified drive
02C6 5F          mov      e,a
02C7      dobdos   drivfn,

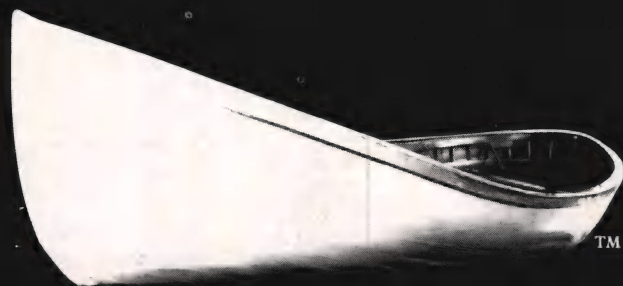
;
02CC      savusr:  dobdos   userfn,0ffh      ;save user #
02D4 321205      sta      uuser
02D7      dobdos   getaddrfn      ;get extent mask for this drive
02DC 23232323    inx h ! inx h ! inx h ! inx h
02E0 7E          mov      a,m
02E1 324704      sta      EXTMASK
02E4 AF          xra      a
02E5 321005      sta      count
02E8 3A6D00      lda      fcb2+1
02EB FE58        cpi      'X'
02ED 210D05      lxi      h,flags
02F0      res      MAKPUBBIT,m
02F2      jrz      find
02F4      setb      MAKPUBBIT,m

;
;      PAGE
;      find all filename entries in all user numbers
;
02F6 215C00      find:  lxi      h,fcbl
02F9 363F        mvi      m,'?'      ;match ALL dir entries
02FB EB          xchg
02FC      dobdos   srchfstfn,
0301 320E05      sta      indx      ;save position in buffer
0304 3C          inr      a

```

(Continued on page 66)

*C Is The Language.
Lifeboat Is The Source.*



*Lifeboat.TM
The Leading Source And Authority For Serious Software.
1-800-847-7078.*

In NY State: 212-860-0300

Serious Software For The C Programmer From Lifeboat.™

Lattice® C Compiler: *The serious software developer's first choice.*

Selected for use by IBM,® Texas Instruments, Wang,® MicroPro,® Ashton-Tate,™ IUS/Sorcim,® Microsoft® and Lotus™ to name a few of the many. Why?

Lattice C is clearly the finest 16 bit C compiler available today.

- Renowned for speed and code quality.
- Fully compatible with the C standards set forth by Kernighan and Ritchie.
- Four memory model options offer you unsurpassed control and versatility.
- Superior quality documentation.
- Now includes automatic sensing and use of the 8087 chip.
- Widest selection of supporting add-on packages.

Halo™: *A graphics development package rapidly emerging as the industry standard.*

- 140 graphics commands including plot, line, arc, box circle and ellipse primitives, bar and pie charts; pattern fill and dithering commands.
- New: multiple viewports and "stroke text" for angling, scaling and filling text.

C Food Smorgasbord™: *This beautifully written collection of C functions is a valuable time saver.*

- Library includes a binary coded decimal arithmetic package, level 0 I/O functions, a terminal independence package, IBM PC ROM BIOS access functions and much more.

Pmate™: *The premier editor for the programming professional.*

Pmate is a full screen editor with its own powerful macro command language:

- Perform on screen row and column arithmetic, alphabetize lists, translate code from one language to another, call up other macros.
- Customize Pmate almost any way you like.
- Contains 10 auxiliary buffers for storage of macros, text, subroutines.
- An "undo" feature allows the programmer to retrieve whole series of deleted items.

Additional C Tools

Available From Lifeboat:

Panel™: Screen formatter and data entry aid.

Lattice Windows™: Windowing utility; create "Virtual Screens."

Plink-86™: The popular linker; includes extensive overlay capabilities.

Pfix86™: Dynamic debugging utility.

Pfix86 Plus™: Symbolic debugger with capacity to debug overlays.

Btrieve™: Database record access/retrieval library.

Phact: Multikeyed ISAM C-Function library.

Fabs: Fast access B-tree database function library.

Autosort: Fast sort/merge utility.

ES/P: 'C' program entry with automatic syntax checking and formatting.

Greenleaf Functions™: Library of over 200 popular C functions.

And much more.

YES! Please rush me the latest FREE Lifeboat™ catalog of C products.

Company
Name

Business
Phone

Name

Title

Address

City

State

Zip

Please check the category where Lifeboat can best help you:

☐ Software development

☐ Corporate

☐ Education

☐ Dealer/distributor

☐ Government

☐ Other

Call Direct: 1-800-847-7078 (In NY State: 212-860-0300)

Return coupon to: Lifeboat Associates™

1651 Third Avenue, New York, NY 10128.

DD/11

Listing Two

```

0305                jrz     done                ;no entries at all
;
0307 CD1604 findall:call    chknxt                ;is entry PUBLIC or specified filename?
030A                jrnz    findnxt
030C CD5304        call    savefcb                ;yes - save it
030F CD7304        call    setcol
0312 CD8D04        call    printentry            ; list it
0315 211005        lxi     h,count                ; and count it
0318 34            inr     m
0319 findnxt:doebdos srchnxtfn,fcb
0321 320E05        sta     indx
0324 3C            inr     a
0325                jrnz    findall
;
0327 210D05 nomore: lxi     h,flags
032A                bit     LISTPUBBIT,m
032C 3A1005        lda     count
032F                jrz     nom0
0331 B7            ora     a
0332                jrnz    done
0334                print   nopubs
033A                jr      done
;
033C D601 nom0:   sui     1
033E                jrc     none
0340                jrnz    nochanges
0342 210D05        lxi     h,flags                ;exactly 1 file found
0345                bit     MAKPUBBIT,m
0347                jrz     noml
0349 CD9003        call    setpub
034C                jr      done
034E CDC803 noml:  call    setpriv
0351                jr      done
;
                PAGE
nochanges:        ;can't be PUBLIC if > 1 match on drive
0353                print   multimg
0359                jr      done
;
035B none:   print   nonemsg
0361                jr      done
;
0363 110902 nopub: lxi     d,pubnam
0366 CD0C04        call    cant
;fall thru
;
;    all done, restore drive/user and return
;
0369 210D05 done:  lxi     h,flags
036C                bit     LISTPUBBIT,m
036E                jrz     done0
0370                print   usage
0376 3A1105 done0: lda     udrive ;relogin user's drive
0379 5F            mov     e,a

```



```

037A          dobdos  drivefn,
037F 3A1205   donel:  lda    uuser
0382 5F       mov    e,a
0383          dobdos  userfn,
0388 CDBF04   call   crlf
038B          xit:    lspd   ustack
038F C9       ret

;
;       set the PUBLIC attribute bit
;
0390 211705   setpub: lxi    h,pubfcb+2      ;test attr bit 2
0393          bit     7,m
0395          jrnz    ispub                  ;quit if already PUBLIC
0397          setb    7,m                    ;set attr bit 2
0399 3A1505   lda     pubfcb                  ;save user # for output
039C 5F       mov    e,a                      ;set user # of this file
039D          dobdos  userfn,
03A2 211505   lxi     h,pubfcb
03A5 3600     mvi     m,0                      ;put default drive into fcb
03A7 EB       xchg
03A8          dobdos  setattrfn,
03AD 3C       inr    a
03AE          jrz     nopub
03B0          print   tomsg
03B6          print   pubnam

setpend:
03BC 211505   lxi     h,pubfcb
03BF C39004   jmp     printentl

;
;
;       reset the PUBLIC attribute bit
;
setpriv:
03C8 3A1505   lda     pubfcb                  ;save file user #
03CB 5F       mov    e,a                      ;set user # of fcb
03CC          dobdos  userfn,
03D1 211705   lxi     h,pubfcb+2              ;reset PUBLIC attr bit
03D4          bit     7,m
03D6          jrz     ispriv                  ;quit if it's already private
03D8          res     7,m
03DA 211505   lxi     h,pubfcb
03DD 3600     mvi     m,0                      ;default drive
03DF EB       xchg
03E0          dobdos  setattrfn
03E5 3C       inr    a
03E6          jrz     nopriv
03E8          print   tomsg
03EE          print   privnam
03F4          jr      setpend

;
;       PAGE
03F6 111302   nopriv: lxi     d,privnam
03F9 C30C04   jmp     cant

;
03FC 111302   ispriv: lxi     d,privnam
;             fall thru

```

(Continued on next page)

Listing Two

```

03FF D5      istype: push    d
0400          print    ismsg
0406 D1      ;      fall thru
;
;
printde:          ;bdos string print function
0407 0E09      mvi      c,9
0409 C30500    jmp      5
;
040C D5      cant:   push    d
040D          print    cantmsg
0413 D1      pop     d          ;print 2nd msg
0414          jr      printde
;
;      check next directory entry
;      if listing PUBLIC files, ret Z if PUBLIC and 0th extent
;      if matching a filename, ret Z if same name,type and extent
;      else ret NZ
0416 CD6304    chknxt: call    findentry
0419 7E      mov     a,m
041A FEE5      cpi     0E5h          ;don't match erased entries
041C          jrnz    chkn0
041E B7      ora     a          ;set nz
041F C9      ret
0420 23      chkn0:  inx     h
0421 3A0D05    lda     flags
0424          bit     LISTPUBBIT,a
0426          jrz     chkn1
;
;      list all PUBLIC files
0428 23      inx     h          ;point at 2nd char of filename
0429 7E      mov     a,m
042A 2F      cma
042B E680      ani     80h          ;check compl of attr bit
042D C0      rnz          ;not PUBLIC - ret NZ
042E 110A00    lxi     d,12-2      ;have a PUBLIC file,
0431 19      dad     d          ;point at its extent byte
0432 AF      xra     a          ;and check for extent 0
0433          jr      chkn3
;
;      check for match with specified filename/extent 0
;note* doesn't allow wild cards
0435 115D00    chkn1:  lxi     d,fcbl
0438 060B      mvi     b,11          ;name & type
043A 1A      chkn2:  ldax    d
043B 96      sub     m
043C E67F      ani     7fh          ;don't test attr bits
043E 23      inx     h
043F 13      inx     d
0440 C0      rnz          ;nz if no match
0441          djnz    chkn2
0443 1A      ldax    d          ;now check the extent
0444 4E      chkn3:  mov     c,m

```

(Continued on page 70)

Uniforth

Come to us for your state-of-the-art FORTH needs! Announcing the latest additions to the UNIFORTH family:

16-bit Z8000, 68000, 16032
32-bit 80186, 68000, 16032

Obtain these stock items captured under traditional operating systems, or try our DB16000, Slicer and CompuPro stand-alone versions. Complete compatibility is retained throughout the UNIFORTH product line (from the Commodore 64 to the VAX).

Features include software floating point, video editor, full macro assembler, debugger, decompiler, top-notch documentation, etcetera. Prices start at \$175. Call or write for our free brochure.

Unified Software Systems

P.O. Box 2644, New Carrollton, MD 20784, 301/552-9590

DEC, VAX, PDP, RT-11, RSX-11 (TM) Digital Equipment Corp; CP/M (TM) Digital Research; MSDOS (TM) Microsoft; VIC-20 (TM) Commodore.

SMALL C FOR IBM-PC

Small-C Compiler Version 2.1 for PC-DOS/MS-DOS
Source Code included for Compiler & Library
New 8086 optimizations
Rich I/O & Standard Library

\$40

CBUG SOURCE LEVEL DEBUGGER FOR SMALL C

Break, Trace, and Change variables all on the source level
Source code included

\$40

Datalight

11557 8th Ave. N.E.
Seattle, Washington 98125

(206) 367-1803

ASM or MASM is required with compiler.
Include disk size (160k/320k), and DOS version with order.
VISA & MasterCard accepted. Include card no. & expiration date.
Washington state residents include 7.9% sales tax.
IBM-PC & PC-DOS are trademarks of International Business Machines
MS-DOS is a trademark of Microsoft Corporation.

Circle no. 73 on reader service card.

Circle no. 21 on reader service card.

GTEK INC. (601) 467-8048

EPROM PROGRAMMER

Compatible w/ all RS-232 serial interface port • Auto select baud rate • With or without handshaking • Bidirectional Xon/Xoff and CTS/DTR supported • Read pin compatible ROMS • No personality modules • Intel, Motorola, MCS86, Hex formats • Split facility for 16 bit data paths • Read, program, formatted list commands • Interrupt driven, program and verify real time while sending data • Program single byte, block, or whole EPROM • Intelligent diagnostics discern bad and erasable EPROM • Verify erasure and compare commands • Busy light • Complete w/Textool zero insertion force socket and integral 120 VAC power (240 VAC/50Hz available)

DR Utility Package allows communication with 7128, 7228, and 7956 programmers from the CP/M command line. Source Code is provided. PGX utility package allows the same thing, but will also allow you to specify a range of addresses to send to the programmer. Verify, set the Eprom type.

MODEL 7316 PAL PROGRAMMER
Programs all series 20 PALS. Software included for compiling PAL source codes.

Software Available for CPM,¹ ISIS,² TRSDOS,³ MSDOS.⁴

1. TM of Digital Research Corp.
2. TM of Intel Corp.
3. TM of Tandy Corp.
4. TM of Microsoft.

Post Office Box 289
Waveland, Mississippi 39576
[601]-467-8048

DEVELOPMENT HARDWARE/SOFTWARE

HIGH PERFORMANCE/ COST RATIO

\$879 stand alone MODEL 7956
Intelligent algorithm. Stand alone, copies eight EPROMs at a time. With RS-232 option \$1099.

\$549 MODEL 7316
MODEL 7316 PAL PROGRAMMER
Programs all series 20 PALS. Software included for compiling PAL source codes.

\$549 MODEL 7228
MODEL 7228 EPROM PROGRAMMER
All features of Model 7128 plus Auto Select Baud ... super fast adaptive programming algorithms, low profile aluminum enclosure. Programs 2764 in one minute!

\$429 MODEL 7128
MODEL 7128 EPROM PROGRAMMER
Programs and Reads:

Programs	Reads
NMOS 2508	CMOS 27C16
NMOS 2516	CMOS 27C32
NMOS 2532	CMOS 27C64
NMOS 2564	CMOS 27C128
NMOS 68766	CMOS 27C256
NMOS 68764	CMOS 27C512
NMOS 8755	CMOS 27C1024
NMOS 5133	CMOS 27C2048
EEPROM 5213	EEPROM 5213H
EEPROM 5213H	EEPROM X2816
EEPROM X2816	EEPROM 48016
EEPROM 48016	EEPROM 12816A
MPU'S 8748	MPU'S 8748H
MPU'S 8748H	MPU'S 8749H
MPU'S 8749H	MPU'S 8741
MPU'S 8741	MPU'S 8742H
MPU'S 8742H	MPU'S 8741H
MPU'S 8741H	MPU'S 8751

\$1195 MODEL 7324
MODEL 7324 PAL PROGRAMMER
Programs all series 20 & 24 PALS. Operates stand alone or via RS232.

Avocet Cross Assemblers are available to handle 8748, 8751, Z8, 6502, 680X, etc. Available for CP/M and MSDOS computers. Order by processor type and specify kind of computer.

Model DE-4 U/V Products hold 8, 28 pin parts. High quality professional construction.

Circle no. 31 on reader service card.

Listing Two

```

;
;      check for same extent in A, C
;
SAMEEXT:      ;z set if same extent
;;      push      b      ;;bc not needed
            push      psw
0445 F5
0447 =      EXTMASK equ $+1
0446 3E00      mvi      a,00h
0448 2F      cma
0449 47      mov      b,a      ;save mask
044A 79      mov      a,c      ;mask C
044B A0      ana      b
044C 4F      mov      c,a      ;save in C
044D F1      pop      psw      ;now do A
044E A0      ana      b
044F 91      sub      c
0450 E61F     ani      1Fh      ;check only legal bits 0-4
;
;      pop      b
0452 C9      ret

;
;
savefcb:
0453 CD6304      call     findentry
0456 7E      mov      a,m      ;save the user #
0457 329504      sta      fileuserno
045A 111505      lxi      d,pubfcb
045D 012000      lxi      b,32
0460      ldir
0462 C9      ret

;
findentry:      ;find entry in buf
;point to fcb found
;*32
0463 3A0E05      lda      indx
0466 87      add      a
0467 87      add      a
0468 87      add      a
0469 87      add      a
046A 87      add      a
046B 216505      lxi      h,buf
046E 85      add      l
046F 6F      mov      l,a
0470 D0      rnc
0471 24      inr      h
0472 C9      ret

;
setcol:      lda      count
0473 3A1005      ani      03h
0476 E603      jz      crlf
0478 CABF04      call     twosp
047B CD8304      mvi      c,fence
047E 0E7C      call     charout
0480 CDC604      twosp:  mvi      c,space
0483 0E20      call     charout
0485 CDC604      mvi      c,space
0488 0E20      jmp      charout
048A C3C604

```



```

048D CD6304    printentry:                ;print drive/user/filename
                call    findentry
0490 E5        printentl:                ;print at (hl)
                push    h
0491 CDB704    call    printdrv
0495 =         fileuserno equ $+1
0494 3E00      mvi     a,00h
0496 CDD304    call    printuser
0499 E1        pop     h
                ;fall thru
049A 23        print$fn:                ;print filename.ext
                inx     h
049B 0608      mvi     b,8
049D CDA704    call    prfn
04A0 0E2E      mvi     c,'.'
04A2 CDC604    call    charout
04A5 0603      mvi     b,3
                ;fall thru
04A7 7E        prfn:                    ;print filename char, lowercase if attr bit set
04A8 E67F      mov     a,m
04AA BE        ani     7fh                ;kill attr bit when printing
04AB           cmp     m
04AD F620      jr      prfnl
04AF 4F        prfnl: mov     c,a
04B0 CDC604    call    charout
04B3 23        inx     h
04B4           djnz   prfn
04B6 C9        ret
                ;
04B7 3A0F05    printdrv:                ;print drive
04BA C641      lda     drive
04BC 4F        adi     'A'
04BD           mov     c,a
04BD           jr      charout
                ;
04BF 0E0D      crlf:  mvi     c,CR
04C1 CDC604    call    charout
04C4 0E0A      mvi     c,LF
                ;fall thru
                ;
04C6 E5        charout:                 ;preserve registers
04C7 C5        push    h
04C8 D5        push    b
04C9 59        push    d
04CA 0E02      mov     e,c
04CC CD0500    mvi     c,2
04CF D1        call    5
04D0 C1        pop     d
04D1 E1        pop     b
04D2 C9        pop     h
                ;
04D3 FE0A      printuser:               ;print A as user #
04D5           cpi     10
04D7 F5        jrn     printul
04D8 0E20      push    psw                ;1 space if sgl digit
04DA CDC604    mvi     c,' '
                call    charout

```

(Continued on next page)

Listing Two

```

04DD F1                pop     psw
04DE 6F      printul:mov     l,a
04DF 2600                mvi     h,0
04E1 CDE804            call    printdec
04E4 0E3A                mvi     c,':'
04E6                jr      charout

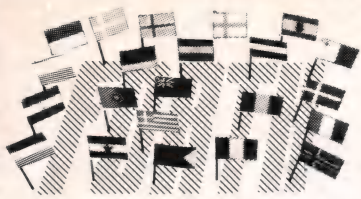
;
printdec:
04E8 F5      DECOUT: PUSH    PSW                ;prints hl in decimal
04E9 C5                PUSH    B
04EA D5                PUSH    D
04EB E5                PUSH    H
04EC 01F6FF            LXI     B,-10
04EF 11FFFF            LXI     D,-1
04F2 09      DECOU2: DAD     B
04F3 13                INX     D
04F4 DAF204            JC      DECOU2
04F7 010A00            LXI     B,10
04FA 09      DAD     B
04FB EB      XCHG
04FC 7C      MOV     A,H
04FD B5      ORA     L
04FE C4E804            CNZ     DECOUT ;recursive
0501 7B      MOV     A,E
0502 C630      ADI     '0'
0504 4F      mov     c,a
0505 CDC604      call    charout
0508 E1      POP     H
0509 D1      POP     D
050A C1      POP     B
050B F1      POP     PSW
050C C9      RET

;
050D 00      flags: db      0
050E 00      indx:  db      0
050F 00      drive: db      0
0510 00      count: db      0
0511 00      udrive: db      0                ;user's drive
0512 00      uuser:  db      0
0513 0000     ustack: dw      0
0515 =      pubfcb equ $
0565 =      stack equ pubfcb +32 + 48

;
0565 =      buf      equ      stack
0515                END

```

End Listings



FORTH into Europe

Support for major FORTHS and our own products

VAX-FORTH 32

- ★ Complete VMS support
- ★ Command line qualifiers
- ★ DEC compatible full screen editor
- ★ On line HELP facilities
- ★ Start-up files
- ★ Switchable log-files
- ★ System files with precompiled modules
- ★ Cross compilers available for most microprocessors

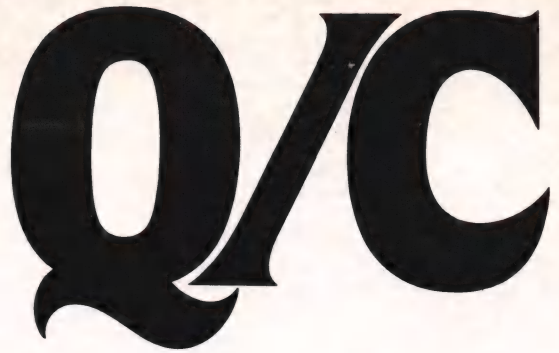
FORTH-83 CROSS-COMPILERS

- ★ B-tree symbol table of unlimited size
- ★ Compiles FORTH-83 nucleus
- ★ Compiles 16 or 32 bit code
- ★ Two passes allow automatic pruning of nucleus for ROM applications
- ★ Automatic handling of defining words
- ★ Targets include 1802, Z8, 8070, 8080, 6801/3, 6502, 6511Q, 6809, 99xxx, 8086/8, 68000, Z80

MicroProcessor Engineering, 21 Hanley Road, Shirley,
Southampton, SO1 5AP, England, Tel: 0703 780084

Die FORTH-Systeme Angelika Flesch, Schuetzenstrasse 3,
7820 Titisee-Neustadt, West Germany, Tel: 07651 1665

Circle no. 47 on reader service card.



For only \$95, Q/C is a ready-to-use C compiler for CP/M. You get complete source code for the compiler and over 75 library functions. Q/C is upward compatible with UNIX Version 7 C, but doesn't support long integers, float, parameterized #defines, and bit fields.

- Full source code for compiler and library.
- No license fees for object code.
- Z80 version takes advantage of Z80 instructions.
- Excellent support for assembly language and ROMs.
- Q/C is standard. Good portability to UNIX.

Version 3.2 of Q/C has many new features: structure initialization, faster runtime routines, faster compilation, and improved ROM support. Yes, Q/C has casts, typedef, sizeof, and function typing. The Q/C User's Manual is available for \$20 (applies toward purchase). VISA and MasterCard welcome.



5266 Hollister
Suite 224
Santa Barbara, CA 93111
(805) 683-1585

Q/C, CP/M, Z80, and UNIX are trademarks of Quality Computer Systems, Digital Research, Zilog, Inc., and Bell Laboratories respectively.

Circle no. 10 on reader service card.

UNPARALLELED
PERFORMANCE
and **PORTABILITY**
in an **ISAM PACKAGE**
at an **UNBEATABLE**
PRICE



c-tree™
BY FAIRCOM

2606 Johnson Drive
Columbia MO 65203

The company that introduced micros to B-Trees in 1979 and created ACCESS MANAGER™ for Digital Research, now redefines the market for high performance, B-Tree based file handlers. With c-tree™ you get:

- complete C source code written to K & R standards of portability
- high level, multi-key ISAM routines and low level B-Tree functions
- routines that work with single-user and network systems
- no royalties on application programs

\$395 COMPLETE

Specify format:
8" CP/M® 5¼" PC-DOS 8" RT-11

for VISA, MC or COD orders, call
1-314-445-6833

Access Manager and CP/M are trademarks of Digital Research, Inc.
c-tree and the circular disc logo are trademarks of FairCom

© 1984 FairCom

Circle no. 27 on reader service card.

A Guide to Resources for the C Programmer

by Terry A. Ward

"This guide to C language resources lists the books available on the language, the articles written concerning C, the producers of C compilers and software, and even an abridged guide to public domain C software."

Furniture: Chippendale
Automobiles: '57 Chevy
Authors: Cain, Hendrix, and Ream

The current rage among the game-playing public is "trivia" games. If we imagined the above three answers to be part of a trivia game, we could probably reconstruct the question of interest. Obviously, "Chippendale" and "'57 Chevy" refer to classic items of furniture and automobiles, respectively. But who or what are Cain, Hendrix, and Ream, and where do they fit in?

Faithful readers of *Dr. Dobb's* will recognize the names and perhaps come to the correct conclusion that they are "classic" authors of public domain and other significant C programs. Namely, they are the authors of a Small-C compiler (Cain), an improved Small-C compiler (Hendrix), and two full-screen text editors (Ream).

To many, though, the names of Cain, Hendrix, and Ream are a mystery. This article is a guide to such C language resources. It lists books available on the language, articles written concerning C, software producers of C compilers, and even an abridged guide to public domain C software.

Prior to compiling this guide, I wrote an article for *Byte* magazine (August 1983) that annotated 100 C articles and books. In the interim, this list has grown to over 200 article citations. The information presented below is available in expanded form in my forthcoming book *Applied Programming Techniques in C* to be published by Scott, Foresman and Company at the end of 1984. The information is current as of September 1984.

The Classics

Our first section of material deals with what might be called the classics of C programming. Just as a real programmer can write a Fortran program in any language, all true C programmers should be familiar with these classics.

On the book front, Kernighan and Ritchie's book *The C Programming Language* (Prentice-Hall, 1978) stands at the pinnacle, defining the very language. Equally classic is *The C Puzzle Book* by Alan C. Feuer. While not a guide to the language, Feuer's book nevertheless provides a novel approach to language understanding. Feuer presents segments of C code (some elegant, some atrocious) and asks "what does it do?" In the process of untangling his puzzles, one learns the C language. For the advanced programmer there are two books of applied techniques: my book from Scott, Foresman and the book by Purdum, Leslie, and Stegemoller.

On the program front are several articles that should be read by all C programmers. The work by Cain created the concept of Small-C. Hendrix has expanded the Small-C compiler and brought it up to date in his work. Finally, Ed Ream

Terry A. Ward, Academic Computing Services, University of Northern Iowa, Cedar Falls, IA 50614.

Portions of this material are excerpted from Applied Programming Techniques in C by Terry A. Ward, published by Scott, Foresman and Company, © 1984, 1900 East Lake Avenue, Glenview, IL 60025.

has created not one but two excellent screen editors for the C programmer. These prime examples of C software all first appeared in the pages of *Dr. Dobb's Journal*.

Books

The books listed below are all valuable to the C programmer. The works by Plum concerning standards and the Bell Laboratory materials are perhaps the most specialized. With the increasing popularity of C, the majority of these books are available in the mass-market bookstores that carry computer-related books and magazines. Those noted by an asterisk (*) are perhaps best suited for the neophyte C programmer.

Bell Laboratories. *Unix Programmer's Manual*. 2 volumes. New York: Holt, Rinehart and Winston, 1983.

*Chirlian, Paul M. *Introduction to C*. Beaverton, OR: Matrix Publishers, Inc., 1984.

Feuer, Alan R. *The C Puzzle Book*. Englewood Cliffs, NJ: Prentice-Hall, 1982.

*Hancock, Les, and Morris Krieger. *The C Primer*. New York: Byte/McGraw-Hill, 1982.

Kernighan, Brian W., and Dennis M. Ritchie. *The C Programming Language*. Englewood Cliffs, NJ: Prentice-Hall, 1978.

*Kochan, Stephen G. *Programming in C*. Rochelle Park, NJ: Hayden Book Company, 1983.

Plum, Thomas. *C Programming Standards and Guidelines: Version U (UNIX and Offspring)*. Cardiff, NJ: Plum Hall, 1982.

Plum, Thomas. *C Programming Standards and Guidelines: Version W (Whitesmith Version)*. Cardiff, NJ: Plum Hall, 1982.

Plum, Thomas. *Learning to Program in C*. Cardiff, NJ: Plum Hall, 1983.

Purdum, Jack. *C Programming Guide*. Indianapolis, IN: Que Corporation, 1983.

Purdum, J., T. Leslie, and A. Stegemoller. *C Programmer's Library*. Indianapolis, IN: Que Corporation, 1983.

*Traister, Robert J. *Programming in C for the Microcomputer User*. Englewood Cliffs, NJ: Prentice-Hall, 1984.

Zahn, C. T. *C Notes: A Guide to the C Programming Language*. New York: Yourdon Press, 1979.

Periodicals/Newsletters

The major periodical sources for C materials are *Dr. Dobb's Journal* and The C Users' Group newsletter publication. Readers of *Dr. Dobb's* are aware that the software in C that has appeared in these pages ranges from C compilers to sophisticated full-screen text editors. The latter group is a source of an informative newsletter and over 15 megabytes of C source code (as of June 1984).

The C Letter. Whitesmith's, Ltd., 97 Lowell Road, Concord, MA 01742.

C Users' Group. C Users' Group, 415 E. Euclid, McPherson, KS 67460.

Dr. Dobb's Journal. M & T Publishing, 2464 Embarcadero Way, Palo Alto, CA 94303.

Microsystems. Ziff Davis Publishing Company, One Park Ave., New York, NY 10016.

Unique. InfoPro Systems, P. O. Box 33, East Hanover, NJ 07936.

Unir Project. Unir Corporation, 5987 E. 71st, Suite 106, Indianapolis, IN 46220.

Unix Review. 2711 76th Avenue, Southeast, Mercer Island, WA 98040.

UNIX/World. Tech Valley Publishing, 289 S. San Antonio Road, Los Altos, CA 94022.

USENIX Assoc. P. O. Box 7, El Cerrito, CA 94530.

/usr/ Group. 4655 Old Ironside Dr., Santa Clara, CA 95054.

World Unix & C. P. O. Box 5314, Mt. Carmel, CT 06518.

C Compilers

The list below categorizes the available C compilers for microcomputers as of September 1984. The addresses for the companies listed are presented alphabetically at the end of this section:

CP/M compilers

Alcor Systems

Alpha Omega Computer Systems

BD Software

Carousel Microtools

The Code Works

Ecosoft

Elfin Systems

ISE-USA

J. E. Hendrix

Kadak Products Limited

Knowology

LSI Japan Co. Ltd.

Manx Software

New Generation Systems

Software Toolworks

Supersoft

Telecon Systems

Western Wares

Whitesmith's

Tandy TRSDOS compilers

Alcor

ISE-USA

IBM PC DOS/MSDOS/CP/M-86

Carousel Microtools

Computer Innovations

Control-C Software

Coriolis Company

c-Systems

C Ware

Datalight

Digital Research

Ecosoft

LanTech Systems

Lifeboat Associates

Manx Software

Mark Williams Co.

Microsoft Corp.

Software Kinetics, Ltd.

Supersoft

Telecon
Unisource Corporation
Whitesmith's

Apple DOS

Manx Software

Unix-like Software Tools

Carousel Microtools
J. E. Hendrix
Knowology
Lifeboat Associates
Mark Williams Co.
New Generation Systems
Whitesmith's

Compiler Addresses

Alcor Systems, 800 W. Garland Avenue, Suite 204, Garland, TX 75040.
Alpha Omega Computer Systems, P. O. Box U, Corvallis, OR 97339.
BD Software, P. O. Box 2368, Cambridge, MA 02238.
Carousel Microtools, 609 Kearney Street, El Cerrito, CA 94530.
The Code Works, 5266 Hollister, Suite 224, Santa Barbara, CA 93111.
Computer Innovations, 980 Shrewsbury Avenue, Suite J504, Tinton Falls, NJ 07724.
Control-C Software, 6441 S.W. Canyon Court, Portland, OR 97221.
The Coriolis Company, P. O. Box 76, Clinton Corners, NY 12514.
c-Systems, P. O. Box 3253, Fullerton, CA 92634.
C Ware, P. O. Box 710097, San Jose, CA 95171.
Datalight, 11557 8th Avenue NE, Seattle, WA 98125.
Digital Research, P. O. Box 579, Pacific Grove, CA 93950.
Ecosoft, 6413 N. College Avenue, Indianapolis, IN 46220.
Elfin Systems, 265 Nogal Drive, Santa Barbara, CA 93110.
InfoSoft, 25 Sylvian Road, South Westport, CT 06880.
Introl Corp., 647 W. Virginia Street, Milwaukee, WI 53204.
ISE-USA, 85 W. Algonquin Road, Suite 400, Arlington Heights, IL 60005.
J. E. Hendrix, Box 8378, University, MS 38677-8378.
Kadak Products Ltd., 206-1847 W. Broadway Avenue, Vancouver, B.C., Canada V6J 1Y5.
Knowology, P. O. Box 283, Wilsonville, OR 97070.
LanTech Systems, 9635 Wendell Road, Dallas, TX 75243.
Lifeboat Associates, 1651 Third Avenue, New York, NY 10028.
LSI Japan Co. Ltd., 2-24-9 Yoyogi, Shibuya-Ku, Tokyo (151), Japan.
Manx Software, Box 55, Shrewsbury, NJ 07701.
Mark Williams Co., 1430 W. Wrightwood Avenue, Chicago, IL 60614.
Microsoft Corp., 10700 Northrup Way, Bellevue, WA 98004.
New Generation Systems, 1800 Michael Faraday Drive, Suite 206, Reston, VA 22090.
Programmer's Shop, 128-M Rockland Street, Hanover, MA 02339.

Software Kinetics, Ltd., 3 Amberwood Crescent, Nepean, Ontario K2E 7L1, Canada.

Software Toolworks, 15233 Ventura Blvd., Suite 1118, Sherman Oaks, CA 91403.

Supersoft, P. O. Box 1628, Champaign, IL 61820.

Telecon Systems, 1155 Meridian Avenue, Suite 218, San Jose, CA 95125.

tiny-C Associates, P. O. Box 269, Holmdel, NJ 07733.

Unipress Software, 2025 Lincoln Highway, Suite 312, Edison, NJ 08817.

Unisource Software, 71 Bent Street, Cambridge, MA 02141.

Vandata, 17544 Midvale Avenue North, Suite 107, Seattle, WA 98133.

Western Wares, Box C, Norwood, CO 81423.

Whitesmith's Ltd., 97 Lowell Road, Concord, MA 01742.

C Software

The firms listed below are the major resources for public domain (or relatively inexpensive non-public-domain) C programs and utilities that include source code.

Algorithmic Technology, Inc., P. O. Box 278, Exton, PA 19341-0278.

Blaise Computing, 2034 Blake Street, Berkeley, CA 94704.

C Source, 12801 Frost Road, Kansas City, MO 64138.

C Users' Group, 415 E. Euclid, McPherson, KS 67460.

Dedicated Micro Systems, 112 N. Main, Box 287, Yates Center, KS 66783.

Dr. Dobb's Journal, M & T Publishing, Inc., 2464 Embarcadero Way, Palo Alto, CA 94303.

Greenleaf Software, 2101 Hickory Road, Carrollton, TX 75006.

J. E. Hendrix, Box 8378, University, MS 38677-8378.

JMI Software Consultants, 1422 Easton Road, Roslyn, PA 19001.

Programmer's Shop, 128-M Rockland Street, Hanover, MA 02339.

Que Corporation, 7960 Castleway Drive, Indianapolis, IN 46250.

Ed Ream, 1850 Summit Avenue, Madison, WI 53705.

Software Toolworks, 15233 Ventura Blvd., Suite 1118, Sherman Oaks, CA 91403.

XOR Corporation, 5421 Opportunity Court, Minnetonka, MN 55343.

Western Wares Software, Box C, Norwood, CO 81423.

C Articles

Listed below are articles and other items from periodicals that relate to C programming, software tools, and utilities written in C. The index provides a quick guide to the basic topics that may be found within the articles.

001 Allison, Dennis. "A Monthly Algorithm Column." *Dr. Dobb's Journal*, 44: 44 - 45 (January 1980).

002 Allison, Dennis. "A Monthly Algorithm Column." *Dr. Dobb's Journal*, 46: 47 - 48 (June/July 1980).

003 Allison, Dennis. "A Monthly Algorithm Column." *Dr. Dobb's Journal*, 48: 44 - 45 (September 1980).

004 Anderson, Bruce. "Type Syntax in the Language C: An

- Object Lesson in Syntactic Innovation." *ACM SIGPLAN Notices*, 15(3): 21 – 27 (March 1980).
- 005 Anderson, Gordon E., and Kenneth C. Shumate. "Selecting a Programming Language, Compiler and Support Environment: Method and Example." *IEEE Computer*, 29 – 36 (August 1982).
- 006 Anonymous. "Review of the C Programming Language (Kernighan & Ritchie)." *Computer Languages*, 4: 199 – 200 (1979).
- 007 Anonymous. "Bell Labs' 32 Bit C/UNIX Micro." *Pipes and Filters*, 1(1): 4 (June 1981).
- 008 Anonymous. "The C Programming Language." *Mini-Micro Software*, 6: n.p. (1981).
- 009 Ashcraft, Steven E. "Ultra Low Level Programming Using a High Level Language." In *Microcomputer Research and Applications: Proceedings of the First Conference of the HP/1000 International Users Group*, Helen K. Brown, ed., 168 – 184 (Elmsford, NY: Pergamon, 1981).
- 010 Azlin, Lawrence A. "A DEBUG Subroutine: A Technique for Making Program Debugging Simpler." *Microsystems*, 100 (December 1983).
- 011 Bailes, P. A. C. "A Co-routine Package for C." *Australian Computer Science Communications*, 1(4): 306 – 309 (December 1979).
- 012 Bailey, Kirk. "Small-C: Bug-Fix Bug." *Dr. Dobb's Journal*, 57: 4 (July 1981).
- 013 Bairstow, Jeffrey. "Getting Started with a New Language." *Personal Computing*, 7(12): 250 (December 1983).
- 014 Baker, Leslie, and Nat Sakowski. "New Improved Latice C." *PC Magazine*, 138 – 141 (March 20, 1984).
- 015 Baker, Leslie, and Nat Sakowski. "Getting Your C-Legs." *PC Magazine*, 118 – 123 (March 20, 1984).
- 016 Barach, David R., and David H. Taenzer. "A Technique for Finding Storage Allocation Errors in C Language Programs." *ACM SIGPLAN Notices*, 17(5): 16 – 23 (May 1982).
- 017 Barach, David R., and David H. Taenzer. "A Technique for Finding Storage Allocation Errors in C Language Programs." *ACM SIGPLAN Notices*, 17(7): 32 – 38 (July 1982).
- 018 Barry, Steve, and Randy Jacobsen. "The TRS-80 Model 16B with Xenix." *Byte*, 288 – 320 (January 1984).
- 019 Bates, Dan. "I Can C Forever." *Dr. Dobb's Journal*, 67: 6 – 7 (May 1982).
- 020 Berenbaum, Alan D., Michael W. Condry, and Priscilla M. Lu. "The Operating System and Language Support Features of the BELLMAC-32 Microprocessor." *ACM SIGPLAN Notices*, 17(4): 30 – 38 (April 1982).
- 021 Bergmann, Ernest E. "PISTOL: A Forth-like Portably Implemented SStack Oriented Language." *Dr. Dobb's Journal*, 76: 12 – 15 (February 1983).
- 022 Birman, H. K., L. N. Rolnitzky, and J. R. Biggee. "A Shape Oriented System for Holter ECG Analysis." In *Computers in Cardiology* (1978).
- 023 Black, Rodney. "Oh Say, Can I C!" *Dr. Dobb's Journal*, 61: 4, 51 (November 1981).
- 024 Bolstad, Terje. "CP/M BDOS and BIOS Calls for C." *Dr. Dobb's Journal*, 80: 22 – 27 (June 1983).
- 025 Bolton, Bill. "Some Useful C Time Functions." *Dr. Dobb's Journal*, 6(8): 16 – 21 (August 1981).
- 026 Bourne, Stephen J. "The Unix Shell." *Byte*, 187 – 204 (October 1983).
- 027 Boyd, Stowe. "Modular C." *ACM SIGPLAN Notices*, 18(4): 48 – 54 (April 1983).
- 028 Brooker, R. A. "A Database Subsystem for BCPL." *Computer Journal*, 25(4): 448 – 464.
- 029 Budd, Timothy C. "An Implementation of Generators in C." *Computer Languages*, 7: 68 – 87 (1982).
- 030 Burkowski, F. J., W. F. Mackey, and M. H. Hamza. "Micro-C: A Universal High Level Language for Microcomputers." *Proceedings of the IEEE International Symposium on Mini and Micro Computers* (Canada/USA, 1977/1978).
- 031 *Byte* (August 1983). The C Language (topical language issue).
- 032 Cain, Ron. "A Small-C Compiler for the 8080's." *Dr. Dobb's Journal*, 45: 5 – 46 (May 1980).
- 033 Cain, Ron. "Runtime Library for the Small-C Compiler." *Dr. Dobb's Journal*, 48: 4 – 15 (September 1980).
- 034 Cain, Ron. "High Praise for Small C." *Dr. Dobb's Journal*, 76: 10 (February 1983).
- 035 Calhoun, Herb. "Malpractice?" *Dr. Dobb's Journal*, 60: 4 – 5 (October 1981).
- 036 Cameron, A.G.W. "The Software Tools Computing Environment." *Microsystems*, 58 – 63 (September 1983).
- 037 Cameron, A.G.W. "The Ratfor Preprocessing Language." *Microsystems*, 52 – 56 (September 1983).
- 038 Cann, Peter. "C Source-Code Formatting." *Byte*, 18 (December 1983).
- 039 Cherry, Lorinda L., and Nina H. Macdonald. "The Unix Writer's Workbench Software." *Byte* (October 1983).
- 040 Christensen, Ward. "The CP/M Users Group Volume 48: Catalogue & Abstracts." *Lifelines*, 1(10): 15 – 16 (March 1981).
- 041 Christensen, Ward. "Full Screen Program Editors for CP/M-80: MINCE." *Lifelines*, 2(11): 7 – 11 (April 1982).
- 042 Christensen, Ward. "Full Screen Program Editors for CP/M-80: Ed Ream's Editor in C." *Lifelines*, 3(5): 43 – 45 (October 1982).
- 043 Christensen, Ward. "MINCE Revisited." *Lifelines*, 3(5): 45 (October 1982).
- 044 Clapp, Douglas. "Microsoft C Unveiled." *PC Magazine*, 503 – 508 (October 1983).
- 045 Clark, David D. "Lmodem: A Small Remote-Communication Program." *Byte*, 410 – 428 (November 1983).
- 046 Colley, William C. "6800 and 1802 Cross-Assemblers for CP/M." *Dr. Dobb's Journal*, 50: 38 – 39 (December 1980).
- 047 Colstad, Ken. "A High-Level Language for Easy X.25 Updates." *Data Communications*, 65 – 77 (September 1981).
- 048 Colvin, James L. "Small-C Compiler." *Dr. Dobb's Journal*, 52: 7, 37 (February 1981).
- 049 Cortesi, D. E. "Dr. Dobb's Clinic." *Dr. Dobb's Journal*, 11 – 19 (November 1983).
- 050 Cotton, G. "A Master Disk Directory." *Interface Age*, 6(11): 104 – 105, 162 – 167 (November 1981).

- 051 Currie, Edward H. "Form over Substance..." *Lifelines* (April 1983).
- 052 Daneliuk, Tim. "LC: A C Compiler for LDOS-Based Machines." *InfoWorld*, 5(25): 53 - 54.
- 053 Darwin, Ian F. "The Unix File." *Microsystems*, 24 - 27 (September 1983).
- 054 Datapro. "An Introduction to the 'C' Language." In *Applications Software Directory* (Delran, NJ: Datapro Research, 1980).
- 055 Dessey, Raymond, ed. "Languages for the Laboratory, Part I." *Analytical Chemistry*, 55(6): 650A - 662A (May 1983).
- 056 Dessey, Raymond, ed. "Languages for the Laboratory, Part II." *Analytical Chemistry*, 55(7): 754A - 764A (June 1983).
- 057 Dietz, Paul F. "Don't Knock C." *Byte*, 576 - 557 (November 1983).
- 058 Ditzel, David R., and H. R. McLellan. "Register Allocation for Free: The C Machine Stack Cache." *ACM SIGPLAN Notices*, 17(4): 48 - 55 (April 1982).
- 059 Dobyns, Barry A. "MINCE: Not Just Another Editor." *Dr. Dobb's Journal*, 6(4): 48 - 52 (April 1981).
- 060 Dobyns, Barry A. "Mincemeat and Minced Words." *Dr. Dobb's Journal*, 56: 4 (June 1981).
- 061 Duncan, Ray. "Tiny-C Interpreter on CDOS." *Dr. Dobb's Journal*, 35: 37 - 39 (September 1979).
- 062 Elliott, Comal. "A Very General Problem-Oriented CAI System." *Behavior Research Methods and Instrumentation*, 14(2): 165 - 169 (April 1982).
- 063 Emerson, Sandra L. "Usenet: A Bulletin Board for Unix Users." *Byte*, 219 - 236 (October 1983).
- 064 Favitta, Michael. "Unica and XM-80." *Dr. Dobb's Journal*, 78: 83 - 85 (April 1983).
- 065 Feuer, Alan R., and Narain H. Gehani. "A Comparison of the Programming Languages C and Pascal." *ACM Computing Surveys*, 14(1): 73 - 92 (March 1982).
- 066 Fiedler, David. "The BDS C Compiler." In *Programmer's Guide to CP/M*, Sol Libes, ed. (Morris Plains, NJ: Creative Computing Press, 1982).
- 067 Fiedler, David. "Unix Facilities on CP/M: Microshell." *Microsystems*, 51 - 53 (January 1983).
- 068 Fiedler, David. "The Unix Tutorial. Part 1: An Introduction to Features and Facilities." *Byte*, 186 - 219 (August 1983).
- 069 Fiedler, David. "The Unix Tutorial. Part 2: Unix as an Applications-Programs Base." *Byte*, 257 - 278 (September 1983).
- 070 Fiedler, David. "The Supersoft C Compiler." *Microsystems* (September 1983).
- 071 Fiedler, David. "The Unix Tutorial. Part 3: Unix in the Microcomputer Marketplace." *Byte* (October 1983).
- 072 Fishman, Harvey. "Janus - A New Ada Compiler for Z80 Systems." *Microsystems*, (6): 70 - 81 (November/December 1982).
- 073 Fitzhorn, Patrick A., and Geatold R. Johnson. "C: Toward a Concise Syntactic Definition." *ACM SIGPLAN Notices*, 16(12): 14 - 21 (December 1981).
- 074 Fitzhorn, Patrick A., and Geatold R. Johnson. "C: Toward a Concise Syntactic Definition: Appendix." *ACM SIGPLAN Notices*, 17(8): 89 - 95 (August 1982).
- 075 Foulk, Richard. "Standard Deviation." *Dr. Dobb's Journal*, 77: 11, 90 (March 1983).
- 076 Garrett, Roger C. "Structured English for the C Programmer." *Interface Age*, 6(10): 30 - 34 (October 1981).
- 077 Garrett, Roger C. "More on C Programming." *Interface Age*, 6(11): 26 - 28, 158 (November 1981).
- 078 Garrett, Roger C. "C.Plus (Conclusion)." *Interface Age*, 6(12): 34 - 38, 142 - 143 (December 1981).
- 079 Gewirtz, David A. "An Introduction to the C Programming Language." *Microsystems*, 2(6): 20 - 38 (November/December 1981).
- 080 Gewirtz, David A. "An Introduction to the C Programming Language (Part II)." *Microsystems*, 3(1): 50 - 58 (January/February 1982).
- 081 Gewirtz, David A. "A Reply to Larry Hamelin." *Microsystems*, 3(4): 12 (July/August 1982).
- 082 Gewirtz, David A. "Two More C Compilers: A Comparative Review of the Aztec C II and C/80 Compiler." *Microsystems*, 83 - 89 (November/December 1982).
- 083 Gibson, T. A., and S. B. Guthery. "Structured Programming, C and tiny-C." *Dr. Dobb's Journal*, 5(5): 30 - 33 (May 1980).
- 084 Gilbreath, Jim. "A High-Level Language Benchmark." *Byte*, 6(9): 180 - 198 (September 1981).
- 085 Gilbreath, Jim, and Gary Gilbreath. "Erasthosthenes Revisited: Once More through the Sieve." *Byte*, 8(1): 283 - 326 (January 1983).
- 086 Gore, Mike, and Bernie Roehl. "Small-C Bug-Fix Bug." *Dr. Dobb's Journal*, 57: 4 - 6, 8 (July 1981).
- 087 Gorman, Walter, and Michael Groussard. "Minicomputer Programming Languages." *ACM SIGPLAN Notices*, 11(4): 4 - 15 (April 1, 1976).
- 088 Hagler, David. "Supersoft Speaks Up." *Byte*, 20 (February 1984).
- 089 Halfant, Matthew. "Small-C for the 9900." *Dr. Dobb's Journal*, 69: 66 - 71 (July 1982).
- 090 Halfant, Matthew. "The Unix C Compiler in a CP/M Environment." *Byte*, 243 - 267 (August 1983).
- 091 Hamelin, Larry. "Intro to the C Programming Language." *Microsystems*, (4): 12 (July/August 1982).
- 092 Hancock, Les. "Growing, Pruning and Climbing Binary Trees with tiny-C." *Dr. Dobb's Journal*, 4(5): 37 - 41, 54 (May 1979).
- 093 Hancock, L. "Implementing a Tiny Interpreter with a CP/M-flavored C." *Dr. Dobb's Journal*, 5(1): 20 - 28 (January 1980).
- 094 Hare, Van Court. "Learning C Inexpensively." *Lifelines*, 30 - 34 (December 1983).
- 095 Harrell, John B. "So Much to C." *80 Micro*, 100 - 106 (February 1984).
- 096 Hatch, William E. "Letter to the Editor." *Microsystems*, 22 (December 1983).
- 097 Hendrix, James, and Ernest Payne. "A New Library for Small-C." *Dr. Dobb's Journal*, 50 - 81 (May 1984) and 56 - 69 (June 1984).
- 097a Hendrix, J. E. "Small VM: Nucleus of a Portable Software Development Environment." *Dr. Dobb's Journal*, 61: 34 - 45 (November 1981).
- 098 Hendrix, J. E. "Small-C Expression Analyzer." *Dr.*

MicroMotion

MasterFORTH

It's here — the next generation of MicroMotion Forth.

- Meets all provisions, extensions and experimental proposals of the FORTH-83 International Standard.
- Uses the host operating system file structure (APPLE DOS 3.3 & CP/M 2.x).
- Built-in micro-assembler with numeric local labels.
- A full screen editor is provided which includes 16 x 64 format, can push & pop more than one line, user definable controls, upper/lower case keyboard entry, A COPY utility moves screens within & between lines, line stack, redefinable control keys, and search & replace commands.
- Includes all file primitives described in Kernigan and Plauger's Software Tools.
- The input and output streams are fully redirectable.
- The editor, assembler and screen copy utilities are provided as relocatable object modules. They are brought into the dictionary on demand and may be released with a single command.
- Many key nucleus commands are vectored. Error handling, number parsing, keyboard translation and so on can be redefined as needed by user programs. They are automatically returned to their previous definitions when the program is forgotten.
- The string-handling package is the finest and most complete available.
- A listing of the nucleus is provided as part of the documentation.
- The language implementation exactly matches the one described in MASTERING FORTH, by Anderson & Tracy. This 200 page tutorial and reference manual is included with MasterFORTH.
- Floating Point & HIRES options available.
- Available for APPLE II/II+ /Ile & CP/M 2.x users.
- MasterFORTH - \$100.00. FP & HIRES - \$40.00 each
- Publications
 - MASTERING FORTH - \$20.00
 - 83 International Standard - \$15.00
 - FORTH-83 Source Listing 6502.Z-80,8086 - \$20.00 each.



Contact:

MicroMotion
12077 Wilshire Blvd., Ste. 506
Los Angeles, CA 90025
(213) 821-4340

Circle no. 46 on reader service card.

FORTRAN PROGRAMMERS

Lahey Computer Systems is pleased to announce

F77L

the complete implementation of the ANSI FORTRAN 77 standard for the IBM PC and compatibles.

With fast compile and execution speeds, specific diagnostics at compile and runtime, F77L meets the needs of mainframe programmers who recognize FORTRAN as the workhorse of computer languages. In addition to meeting the '77 Standard, F77L features include:

- Many IBM H features: \$ in a name, 8 character names. Types: LOGICAL★1, REAL★8, INTEGER★2, COMPLEX★16.
- INCLUDE, OPTION, and CHAIN statements.
- Optional checking: subscript, subprogram class, argument and alternate return count.
- Runtime messages include text and subprogram/line-number traceback.
- COMMONS and subprogram units may be as large as 64K.
- Source file is free format: comments begin with asterisk, continuation lines begin with ampersand.
- Complete and easy to follow 250 page manual.
- Telephone user support and newsletter with updates.

When you purchase the F77L you are buying more than a language system you are also buying LCS's commitment to FORTRAN programming. We have specialized in FORTRAN since 1969 and were the first to implement FORTRAN 77. Because we sell only one product, our customers know that our total effort is directed towards the development of F77L and the continuing service of our users.

If you are serious about FORTRAN programming, then you owe it to yourself to compare Lahey Computer Systems' F77L to the competition.

Complete package: \$477 Visa/MC

To order or for more information call or write:



Lahey Computer Systems, Inc.

904 Silver Spur Road, Suite 417

Rolling Hills Estates, CA 90274 (213) 541-1200

Circle no. 41 on reader service card.



LATTICE® C Compilers

"My personal preferences are Lattice C in the top category for its quick compile and execution times, small incremental code, best documentation and consistent reliability..."

BYTE AUG. 1983

R. Phraner

"...programs are compiled faster by the Lattice C compiler, and it produces programs that run faster than any other C compiler available for PC-DOS."

PC MAGAZINE JULY 1983

H. Hinsch

"...Microsoft chose Lattice C both because of the quality of code generated and because Lattice C was designed to work with Microsoft's LINK program."

PC MAGAZINE OCT. 1983

D. Clapp

"Lattice is both the most comprehensive and the best documented of the compilers. In general it performed best in the benchmark tests."

PERSONAL COMPUTER AGE NOV 1983

F. Wilson

"This C compiler produces good tight-running programs and provides a sound practical alternative to Pascal."

SOFTALK AUG 1983

P. Norton

"...the Lattice compiler is a sophisticated, high-performance package that appears to be well-suited for development of major application programs."

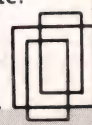
BYTE AUG 1983

Houston, Brodrick, Kent

To order, or for further information
on the LATTICE family of compilers, call or write:



LATTICE, INC.
P.O. Box 3072
Glen Ellyn, IL 60138
(312) 858-7950 TWX 910-291-2150



Circle no. 43 on reader service card.

- Dobb's Journal*, 6(12): 40 – 43 (December 1981).
- 099 Hendrix, J. E. "Small Shell: Part 2 of a North Star VOS." *Dr. Dobb's Journal*, 63: 27 – 35 (January 1982).
 - 100 Hendrix, J. E. "Small-C Compiler, V. 2." *Dr. Dobb's Journal*, 74: 16 – 52 (December 1982).
 - 101 Hendrix, J. E. "Small-C Compiler, V. 2 Continued." *Dr. Dobb's Journal*, 75: 48 – 64 (January 1983).
 - 102 Hendrix, J. E. "Small-C Notes, Bug Fix." *Dr. Dobb's Journal* (January 1983).
 - 103 Hendrix, J. E. "A Slightly Slicker Fix for Small-C." *Dr. Dobb's Journal*, 77: 6 (March 1983).
 - 104 Hinsch, Hanno. "Five C Language Compilers." *PC Magazine*, 210 – 216 (February 1983).
 - 105 Hogan, W. L. "An Evaluation of a Raster Scan Display for Use in an Aircraft Information Handling System." Master's thesis, Naval Postgraduate School, Monterey, CA, 1977.
 - 106 Houston, Jerry, Jim Broderick, and Les Kent. "Comparing C Compilers for CP/M-86." *Byte*, 82 – 106 (August 1983).
 - 107 Houston, Jerry. "Reply to 'Supersoft Speaks Up'." *Byte*, 20 – 21 (February 1984).
 - 108 Howard, Alan D. "Enhancing the C Screen Editor." *Dr. Dobb's Journal*, 79: 38 – 63 (May 1983).
 - 109 Howard, J. E. "An Implementation of a Codasyl Based Data Base Management System under the Unix Operating System." Master's thesis, Naval Postgraduate School, Monterey, CA, 1978.
 - 110 Hughes, Phil. "BASIC, Pascal, or tiny-C? A Simple Benchmarking Comparison." *Byte*, 6(10): 372 – 375 (October 1981).
 - 111 Hunt, Bill. "C and the PC: Part 1." *PC Tech Journal*, 110 – 130 (November/December 1983).
 - 112 Hunt, Bill. "C and the PC: Part 2." *PC Tech Journal*, 91 – 124 (January 1984).
 - 113 Hunter, Bruce. "A Comparison of C Compilers." *Lifelines*, 3(12): 15 – 21 (May 1983).
 - 114 Hunter, Bruce N. "A Review of Microshell—A Unix Like Utility." *Lifelines*, 3(11): 17 – 21 (April 1983).
 - 115 Jackson, T. R. "Letter to the Editor." *Microsystems*, 3(5): 22 – 23 (September/October 1982).
 - 116 Jaeschke, Rex. "Let's C Now ... Part 1, In the Beginning ..." *The DEC Professional*, 42 – 50 (January 1984).
 - 117 Jaeschke, Rex. "Let's C Now ... Part 2, Looping and Testing." *The DEC Professional*, 26 – 38 (March 1984).
 - 118 Jalics, Paul J., and Thomas S. Heines. "Transporting a Portable Operating System: Unix to an IBM Minicomputer." *Communications of the ACM*, 26(12): 1066 – 1072 (December 1983).
 - 119 Johannson, Jan-Henrik. "Arge and Argv for Small-C." *Dr. Dobb's Journal*, 74: 62 – 64 (December 1982).
 - 120 Johnson, S. C. "A Portable Compiler: Theory and Practice." In *Proceedings 5th ACM Conference on Principles of Programming Languages* (1978).
 - 121 Johnson, S. C., and D. M. Ritchie. "Portability of C Programs and the Unix System." *Bell System Technical Journal*, 2021 – 2048 (July/August 1978).
 - 122 Johnson, Stephen C., and Brian W. Kernighan. "The C Language and Models of Systems Programming." *Byte*, 48 – 60 (August 1983).
 - 123 Jolly, George W. "Review of BCPL—The Language and Its Compiler (Martin Richards and Colin Whitby-Strevens)." *Dr. Dobb's Journal*, 79: 70 (May 1983).
 - 124 Joyce, J. "Review of The C Puzzle Book." *ACM Computing Reviews*, 286 (June 1982).
 - 125 Joyce, James. "A C Language Primer. Part 1: Constructs and Conventions in C." *Byte*, 64 – 78 (August 1983).
 - 126 Joyce, James. "A C Language Primer. Part 2: Tool Building in C." *Byte*, 289 – 302 (September 1983).
 - 127 Joyce, James. "A Tour Through the Unix File System." *Byte*, 170 – 182 (October 1983).
 - 128 Katzenelson, Jacob. "Introduction to Enhanced C (EC)." *Software Practice and Experience*, 13(7): 551 – 576 (July 1983).
 - 129 Katzenelson, Jacob. "Higher Level Programming and Data Abstractions—A Case Study Using Enhanced C." *Software Practice and Experience*, 13(7): 577 – 595 (July 1983).
 - 130 Kern, Christopher. "A User's Look at Tiny-C." *Byte*, 4(12): 196 – 206 (December 1979).
 - 131 Kern, Christopher. "Printf for the C Function Library." *Byte*, 6(5): 40 – 434 (May 1981).
 - 132 Kern, Christopher. "The BDS C Compiler." *Byte*, 6(6): 356 – 362 (June 1981).
 - 133 Kern, Christopher. "MINCE: A Text Editor." *Byte*, 150 – 160 (September 1981).
 - 134 Kern, Christopher. "Microshell and Unica: Unix-Style Enhancements for CP/M." *Byte*, 206 – 219 (December 1982).
 - 135 Kern, Christopher. "Supervyz and Organizr: Two Menu-Driven Front Ends for CP/M." *Byte*, 446 – 451 (January 1983).
 - 136 Kern, Christopher. "The Scribble Text Processor." *Byte*, 302 – 310 (February 1983).
 - 137 Kern, Christopher O. "Five C Compilers for CP/M-80." *Byte*, 110 – 130 (August 1983).
 - 138 Kern, Christopher. "More Unix-Style Software Tools for CP/M." *Byte*, 428 – 434 (October 1983).
 - 139 Kernighan, B. "Why Pascal Is not my Favorite Programming Language." *Computer Science Technical Report #100* (Bell Laboratories, July 18, 1981).
 - 140 King, B. "The Flexibility of C." *CP/M Review*, 1(2): 22 – 23, 75 (January/February 1983).
 - 141 Krieger, M. S., and P. J. Plauser. "C Language's Grip on Hardware Makes Sense for Small Computers." *Electronics* (May 8, 1980).
 - 142 Krieger, Mark, and Fred Pack. "Unix as an Application Environment." *Byte*, 209 – 214 (October 1983).
 - 143 Kvaleberg, Egil. "Small-C DISKDOC: A Repair and Maintenance Utility." *Dr. Dobb's Journal*, 66: 26 – 34 (April 1982).
 - 144 Kvaleberg, Egil. "Doctoring DISKDOC." *Dr. Dobb's Journal*, 67: 6 – 7 (May 1982).
 - 145 Leas, Dennis, and Paul Wintz. "Chisel Your Code with a Profiler." *Byte*, 286 – 290 (August 1983).
 - 146 Lee, P. A. "Exception Handling in C Programs." *Software Practice and Experience*, 13(1983): 389 – 405.
 - 147 Libes, Don. "Reply to T. R. Jackson." *Microsystems*, 3(5): 23 – 26 (September/October 1982).

Dr. Dobb's Journal

Your primary job function: (Check one only)

- ☐ Company management (Pres., V.P., Treas., Owner, Gen. Mgr., Mktg. Dir.)
- ☐ Computer systems management (V.P. EDP, MIS Director, Data Processing Mgr., Data Communications Mgr., Network Planner)
- ☐ Engineering management (V.P. Engr., Chief Engr., Tech. Director, Dir. R&D)
- ☐ Systems integrators (Systems Designer, Project Engr., Systems Application Engr., Technical Staff Members)
- ☐ Consultants (Computer/EDP/Data Communications Consultant)
- ☐ Educators (Educational users and instructors of computer technology)
- ☐ Systems/Programming specialists—mini-micro systems
- ☐ Other (Please specify) _____

B. Which languages are you MOST interested in?

- | | | |
|-----------------------------------|---------------------------------|--------------------------------------|
| <input type="checkbox"/> BASIC | <input type="checkbox"/> C | <input type="checkbox"/> PL/I |
| <input type="checkbox"/> Fortran | <input type="checkbox"/> LISP | <input type="checkbox"/> APL |
| <input type="checkbox"/> COBOL | <input type="checkbox"/> Prolog | <input type="checkbox"/> Logo |
| <input type="checkbox"/> Pascal | <input type="checkbox"/> Ada | <input type="checkbox"/> Smalltalk |
| <input type="checkbox"/> Modula-2 | <input type="checkbox"/> Forth | <input type="checkbox"/> Other _____ |

C. What is the operating system?

- ☐ CP/M (or derived)
- ☐ UNIX (or derived)
- ☐ MS-DOS (or derived)
- ☐ Other _____

Nov. 1984, No. 97

Dr. Dobb's Journal

A. Your primary job function: (Check one only)

- ☐ Company management (Pres., V.P., Treas., Owner, Gen. Mgr., Mktg. Dir.)
- ☐ Computer systems management (V.P. EDP, MIS Director, Data Processing Mgr., Data Communications Mgr., Network Planner)
- ☐ Engineering management (V.P. Engr., Chief Engr., Tech. Director, Dir. R&D)
- ☐ Systems integrators (Systems Designer, Project Engr., Systems Application Engr., Technical Staff Members)
- ☐ Consultants (Computer/EDP/Data Communications Consultant)
- ☐ Educators (Educational users and instructors of computer technology)
- ☐ Systems/Programming specialists—mini-micro systems
- ☐ Other (Please specify) _____

B. Which languages are you MOST interested in?

- | | | |
|-----------------------------------|---------------------------------|--------------------------------------|
| <input type="checkbox"/> BASIC | <input type="checkbox"/> C | <input type="checkbox"/> PL/I |
| <input type="checkbox"/> Fortran | <input type="checkbox"/> LISP | <input type="checkbox"/> APL |
| <input type="checkbox"/> COBOL | <input type="checkbox"/> Prolog | <input type="checkbox"/> Logo |
| <input type="checkbox"/> Pascal | <input type="checkbox"/> Ada | <input type="checkbox"/> Smalltalk |
| <input type="checkbox"/> Modula-2 | <input type="checkbox"/> Forth | <input type="checkbox"/> Other _____ |

C. What is the operating system?

- ☐ CP/M (or derived)
- ☐ UNIX (or derived)
- ☐ MS-DOS (or derived)
- ☐ Other _____

Reader Service Card

D. Please indicate which of the following microcomputers you currently own and/or plan to buy in the next 12 months.

	Own	Plan to Buy
Apple	<input type="checkbox"/>	<input type="checkbox"/>
Commodore	<input type="checkbox"/>	<input type="checkbox"/>
Digital Equipment/DEC	<input type="checkbox"/>	<input type="checkbox"/>
Heath/Zenith	<input type="checkbox"/>	<input type="checkbox"/>
Hewlett-Packard	<input type="checkbox"/>	<input type="checkbox"/>
IBM	<input type="checkbox"/>	<input type="checkbox"/>
Macintosh	<input type="checkbox"/>	<input type="checkbox"/>
Radio Shack/Tandy TRS 80	<input type="checkbox"/>	<input type="checkbox"/>
Texas Instruments	<input type="checkbox"/>	<input type="checkbox"/>
Other (Specify)	<input type="checkbox"/>	<input type="checkbox"/>
None	<input type="checkbox"/>	<input type="checkbox"/>

E. What best describes the work you do with this microcomputer?

- ☐ Business functions
- ☐ Software/Hardware development
- ☐ Scientific/Engineering/R&D applications

F. Are you the decision maker or very influential in computer-related purchases at work?

- ☐ Yes ☐ No

G. Is your company a dealer, distributor, or systems house for microcomputers?

- ☐ Yes ☐ No

H. Subscriber

- ☐ Yes ☐ No

To obtain information about products or services mentioned in this issue, circle the appropriate number listed below. Use bottom row to vote for best article in issue. One card per person.

1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	24	25	26	2	
28	29	30	31	32	33	34	35	36	37	38	39	40	41	42	43	44	45	46	47	48	49	50	51	52	53	5
55	56	57	58	59	60	61	62	63	64	65	66	67	68	69	70	71	72	73	74	75	76	77	78	79	80	8
82	83	84	85	86	87	88	89	90	91	92	93	94	95	96	97	98	99	100	101	102	103	104	105	106	107	10
109	110	111	112	113	114	115	116	117	118	119	120	121	122	123	124	125	126	127	128	129	130	131	132	133	134	13

Articles: 190 191 192 193 194 195 196 197 198 199 200 201 202 203 204 205 206 207 208 209 210 211 212 213 214 215 216 217 218 219 220 221 222 223 224 225 226 227 228 229 230 231 232 233 234 235 236 237 238 239 240 241 242 243 244 245 246 247 248 249 250 251 252 253 254 255 256 257 258 259 260 261 262 263 264 265 266 267 268 269 270 271 272 273 274 275 276 277 278 279 280 281 282 283 284 285 286 287 288 289 290 291 292 293 294 295 296 297 298 299 300 301 302 303 304 305 306 307 308 309 310 311 312 313 314 315 316 317 318 319 320 321 322 323 324 325 326 327 328 329 330 331 332 333 334 335 336 337 338 339 340 341 342 343 344 345 346 347 348 349 350 351 352 353 354 355 356 357 358 359 360 361 362 363 364 365 366 367 368 369 370 371 372 373 374 375 376 377 378 379 380 381 382 383 384 385 386 387 388 389 390 391 392 393 394 395 396 397 398 399 400 401 402 403 404 405 406 407 408 409 410 411 412 413 414 415 416 417 418 419 420 421 422 423 424 425 426 427 428 429 430 431 432 433 434 435 436 437 438 439 440 441 442 443 444 445 446 447 448 449 450 451 452 453 454 455 456 457 458 459 460 461 462 463 464 465 466 467 468 469 470 471 472 473 474 475 476 477 478 479 480 481 482 483 484 485 486 487 488 489 490 491 492 493 494 495 496 497 498 499 500 501 502 503 504 505 506 507 508 509 510 511 512 513 514 515 516 517 518 519 520 521 522 523 524 525 526 527 528 529 530 531 532 533 534 535 536 537 538 539 540 541 542 543 544 545 546 547 548 549 550 551 552 553 554 555 556 557 558 559 560 561 562 563 564 565 566 567 568 569 570 571 572 573 574 575 576 577 578 579 580 581 582 583 584 585 586 587 588 589 590 591 592 593 594 595 5

NO POSTAGE
NECESSARY
IF MAILED
IN THE
UNITED STATES

BUSINESS REPLY MAIL

FIRST CLASS PERMIT #27346, PHILADELPHIA, PA.

POSTAGE WILL BE PAID BY ADDRESSEE

SOFTWARE TOOLS FOR ADVANCED PROGRAMMERS

Dr. Dobb's Journal

P.O. BOX 13851

PHILADELPHIA, PA 19101

NO POSTAGE
NECESSARY
IF MAILED
IN THE
UNITED STATES

BUSINESS REPLY MAIL

FIRST CLASS PERMIT #27346, PHILADELPHIA, PA.

POSTAGE WILL BE PAID BY ADDRESSEE

SOFTWARE TOOLS FOR ADVANCED PROGRAMMERS

Dr. Dobb's Journal

P.O. BOX 13851

PHILADELPHIA, PA 19101

- 148 Libes, Don. "Unix and CP/M." *Microsystems*, 26 – 34 (January 1983).
- 149 Libes, Don. "Unix on Microcomputers." *Microsystems*, 42 – 44 (January 1983).
- 150 Linhart, Jason. "Managing Software Development with C." *Byte*, 172 – 182 (August 1983).
- 151 MacPherson, Andrew. "Catching Bugs in Small-C." *Dr. Dobb's Journal*, 81: 6 – 7 (July 1983).
- 152 Madden, J. Gregory. "C: A Language for Microprocessors?" *Byte*, 2(10): 130 – 138 (October 1977).
- 153 Magenat-Thalmann, Nadia. "Choosing an Implementation Language for Automatic Translation." *Computer Languages*, 7(1982): 161 – 170.
- 154 Mark of the Unicorn. MINCE text editor documentation. (Mark of the Unicorn, 1981).
- 155 Mateti, Prabhaker. "Pascal Versus C: A Subjective Comparison." In *Proceedings of the Symposium on Language Design and Programming Methodology*, 37 – 69 (Sydney, September 10 – 11, 1979).
- 156 Matthews, M. M. "Review: The C Primer." *ACM Computing Reviews*, 24(9): 384 (September 1983).
- 157 McCall, Caddy. "Microshell and Microtools." *CP/M Review*, 26 – 35 (November/December 1983).
- 158 McClure, Robert L. "Likes to C It Standard, Too." *Dr. Dobb's Journal*, 79: 7 (May 1983).
- 158a McDermott, Edward. "Optimizing Strings in C." *Dr. Dobb's Journal*, 18 – 23 (April 1984).
- 159 McKeon, Brian. "A Small-C Operating System." *Dr. Dobb's Journal*, 77: 36 – 61 (March 1983).
- 160 McSkimin, J. R. "REDAS—A Relational Data Access System for Real-Time Applications." In *Proceedings of COMPSAC 1978* (Computer Software and Applications Conference, 1978).
- 161 Meissner, Michael. "Letter to the Editor: Reply to Fitzhorn and Johnson." *ACM SIGPLAN Notices*, 17(8): 84 – 88 (August 1982).
- 162 Mohler, Lorin S. "A Disk Alignment Routine." *Microsystems*, 2(6): 70 (November/December 1981).
- 163 Murphy, Walter V. "RECLAIM: A File Reclamation Utility for Destroyed Directories." *Dr. Dobb's Journal*, 78: 14 – 23 (April 1983).
- 164 Ness, David, and A. Krigman. "MINCE Editor from Mark of the Unicorn." *Infoworld* (May 11, 1981).
- 165 Norris, Bill. "C-Bits (All About BDS C Version 1.45)." *Lifelines*, 2(9): 37 – 38 (February 1982).
- 166 Nowell, Scott. "Cross Check." *Microsystems*, 98 – 102 (September 1983).
- 167 Petersen, Holger. "Review of the BDS C Compiler." *Dr. Dobb's Journal*, 47: 49 – 50 (September 1980).
- 168 Phraner, Ralph A. "Nine C Compilers for the IBM PC." *Byte*, 134 – 168 (August 1983).
- 169 Plauser, P. J. "Review of *The C Programming Language*." *ACM Computing Reviews*, 2 – 4 (January 1979).
- 170 Pournelle, Jerry. "User's Column . . . MINCE is Not Complete EMACS . . ." *Byte*, 294, 298, 300 (July 1982).
- 171 Pournelle, Jerry. "User's Column . . . Kernighan's Lament . . . There's a New C A'Comin . . ." *Byte*, 226 – 236 (December 1982).
- 172 Pournelle, Jerry. "User's Column . . . There's a New C A'Comin . . ." *Byte*, 230, 235, 236 (December 1982).
- 173 Pournelle, Jerry. "The Debate Goes On . . ." *Byte*, 324 (August 1983).
- 174 Pournelle, Jerry. "User's Column . . . BDS C." *Byte*, 88 – 89 (January 1984).
- 175 Pugh, T. "MCALL-C: A Communications Protocol for Personal Computers." *Dr. Dobb's Journal*, 5(2): 16 – 20 (June/July 1980).
- 176 Pugh, T. "BDS C, A Full Compiler from Lifeboat Associates." *Infoworld* (March 31, 1980).
- 177 Ream, Edward K. "A Portable C Screen-Oriented Editor." *Dr. Dobb's Journal*, 63: 18 – 61 (January 1982).
- 178 Ream, Edward K. "Screen-Oriented Bugs." *Dr. Dobb's Journal*, 67: 4 – 7 (May 1982).
- 179 Ream, Edward K. "RED: A Better C Screen Editor, Part I." *Dr. Dobb's Journal*, 81: 34 – 65 (July 1983).
- 180 Ream, Edward. "RED: A Better C Screen Editor, Part II." *Dr. Dobb's Journal*, 82: 62 – 97 (August 1983).
- 181 Reed, Adam. "An Underline Filter for Matrix Printers." *Byte*, 7(3): 300 – 306 (March 1982).
- 182 Reid, Larry, and Andrew P. McKinlay. "Whitesmith's C Compiler." *Byte*, 8(1): 330 – 344 (January 1983).
- 183 Reitz, Randy. "Small-VOS and Small-Tools." *Microsystems*, 4(1): 66 – 69 (January 1983).
- 184 Rifkin, Edward M., and Steve Williams. "The C Language: Key to Portability." *Computer Design*, 143 – 150 (August 1983).
- 185 Ritchie, D. M., S. C. Johnson, M. E. Lesk, and B. W. Kernighan. "The C Programming Language." *Bell System Technical Journal*, 57(6): 1991 – 2019 (July/August 1978).
- 186 Ritchie, D. M., S. C. Johnson, M. E. Lesk, and B. W. Kernighan. "The C Programming Language." *Dr. Dobb's Journal*, 5(5): 20 – 29 (May 1980).
- 187 Ritchie, D. M., S. C. Johnson, M. E. Lesk, and B. W. Kernighan. "The C Programming Language." Reprinted from *Bell System Technical Journal* (July/August 1978). In *Programming Languages: A Grand Tour*, Ellis Horwitz, ed. (Computer Science Press, 1983).
- 188 Roberts, Bruce. "C." *Popular Computing*, 111 – 119 (September 1983).
- 189 Robertson, M. D. "An Extended BASIC Compiler with Graphics Interface for the PDP-11/50 Computer." Master's thesis, Naval Postgraduate College, Monterey, CA, 1977.
- 190 Roth, Richard. "Small C Grows Up." *Dr. Dobb's Journal*, 61: 46 – 50 (November 1981).
- 191 Roth, Richard L. "C Language Prints Multiple Text Lines." *Electronics Design News*, 171 (August 4, 1982).
- 192 Rovegno, H. D. "Using C Language for Microprocessors." *Electro/77 Conference Record* (1977).
- 193 Rovegno, H. D. "A Support Environment for MAC-8 Systems." *Bell Systems Technical Journal*, 57(6; pt. 2): 2251 – 2264 (July/August 1978).
- 194 Runyon, John. "Review of C Notes: A Guide to the C Programming Language." *DEC Professional*, 22 – 26 (November 1982).
- 195 Saloman, F. A. "Software Development for Microprocessors—A Case Study." *Proceedings of COMPSAC 1978* (Computer Software and Applications Confer-

- ence, 1978).
- 196 Scherrer, Deborah, K. Philip, H. Scherrer, Thomas H. Strong, and Samuel J. Penny. "The Software Tools: Unix Capabilities on Non-Unix Systems." *Byte*, 430 – 446 (November 1983).
 - 197 Schreiner, Axel T. "cc—A driver for a Small-C Programming System." *Dr. Dobb's Journal*, 40 – 55 (June 1984).
 - 198 Schreiner, Axel T. "p—A Small-C Preprocessor." *Dr. Dobb's Journal*, 46 – 82 (July 1984).
 - 199 Sethi, Ravi. "A Case Study in Specifying the Semantics of a Programming Language." In *Proceedings of the 7th ACM Symposium on Principles of Programming Languages*, 117 – 130 (Las Vegas, NV, January 28 – 30, 1980) (ACM, 1980).
 - 200 Skjellum, Anthony. "Argum—A C Command Line Processor." *Dr. Dobb's Journal*, 10 – 31 (August 1982).
 - 201 Skjellum, Anthony. "Using C Instead of Assembly Language." *Microsystems*, 33 – 36 (September/October 1982).
 - 202 Skjellum, Anthony. "Expanding Wildcards Under Unix." *Dr. Dobb's Journal*, 73: 12 – 16, 43 (November 1982).
 - 203 Skjellum, Anthony. "UNICA: A Unix-like Utility System for CP/M." *Microsystems*, 59 – 56 (January 1983).
 - 204 Skjellum, Anthony. "C/Unix Programmer's Notebook." *Dr. Dobb's Journal*, 14 – 17 (December 1983).
 - 205 Skjellum, Anthony. "C/Unix Programmer's Notebook." *Dr. Dobb's Journal*, 94 – 97 (February 1984).
 - 206 Skjellum, Anthony. "C/Unix Programmer's Notebook." *Dr. Dobb's Journal*, 94 – 96 (April 1984).
 - 207 Skjellum, Anthony. "C/Unix Programmer's Notebook." *Dr. Dobb's Journal*, 86 – 106 (June 1984).
 - 208 Skjellum, Anthony. "C/Unix Programmer's Notebook." *Dr. Dobb's Journal*, 116 – 120 (August 1984).
 - 209 Skjellum, Anthony. "C/Unix Programmer's Notebook." *Dr. Dobb's Journal*, 116 – 122 (September 1984).
 - 210 Springer, Allen. "A Comparison of Language C and Pascal." *Technical Report G320-2128* (IBM Cambridge Scientific Center, Cambridge, MA, 1979).
 - 211 Stallings, S. "C into the Future." *PC Magazine*, 142 – 147 (March 20, 1984).
 - 212 Stankowski, J. B. "The Design and Implementation of a General Purpose Graphics Subroutine Library." Master's thesis, Naval Postgraduate College, Monterey, CA, 1976.
 - 213 Staneff, John E. "A Small-C Concordance Generator." *Dr. Dobb's Journal*, 86 – 105 (August 1984).
 - 214 Staneff, John. "A Small-C Help Facility." *Dr. Dobb's Journal*, 84: 40 – 69 (October 1983).
 - 215 Stroustrup, Bjarne. "Classes: An Abstract Data Type Facility for the C Language." *ACM SIGPLAN Notices*, 7(1): 42 – 51 (January 1982).
 - 216 Stroustrup, Bjarne. "Adding Classes to the C Language." *Software Practice and Experience*, 13(1983): 139 – 161.
 - 217 Suckow, Harry. "We'll C You and Raise You ..." *Dr. Dobb's Journal*, 8 – 9, 12 – 16 (January 1984).
 - 218 Taylor, Jeff. "LIST—A Source-Listing Program for the C Language." *Byte*, 6(6): 234 – 246 (June 1981).
 - 219 Taylor, Jeffrey L. "Cross-Reference Generator in C: A Program Conversion Aid." *Dr. Dobb's Journal*, 68: 50 – 54 (June 1982).
 - 220 Taylor, Jeffrey L. "CREF Update." *Dr. Dobb's Journal*, 79: 7, 9 (May 1983).
 - 221 Terry, Chris. "MINCE—A New Text Editor." *Microsystems*, 76 – 79 (May/June 1982).
 - 222 Terry, Chris. "Book Review: *The C Primer, Learning to Program in C*, and *C Programming Guide*." *Microsystems*, 120 (September 1983).
 - 223 Thomas, Rebecca. "What is a Software Tool?" *Byte*, 222 – 238 (August 1983).
 - 224 Tilson, Michael. "Moving Unix to New Machines." *Byte*, 266 – 276 (October 1983).
 - 225 Torkildson, Chris L. "Micro Compiler in Brobdingnag." *Dr. Dobb's Journal*, 72: 6 (October 1982).
 - 226 Tuthill, Bill. "Typesetting on the Unix System." *Byte*, 253 – 262 (October 1983).
 - 227 Van Zandt, James R. "Micro Compiler in Brobdingnag." *Dr. Dobb's Journal*, 72: 6, 59 – 60 (October 1982).
 - 228 Watson, Ron. "Learning the C Language with C-System's C Compiler." *Lifelines*, 4(6): 7 – 8 (November 1983).
 - 229 Watson, Ron. "Software Notes: C-Systems C-Window." *Lifelines*, 26 (January 1984).
 - 230 Wilkinson, James B. "YACC is Alive and Well and Running in RSTS." *VAX/RSTS Professional*, 56 – 61 (October 1983).
 - 231 Willman, Bryan M. "C Clearly With Descriptive Operators." *Dr. Dobb's Journal*, 82: 9 (August 1983).
 - 232 Wilson, Dale. "More on Binary Magic Numbers." *Dr. Dobb's Journal*, 70 – 79 (March 1984).
 - 233 Woods, P. L. "Small-C: An Implementor's Notes and a Bug Corrected." *Dr. Dobb's Journal*, 52: 20 – 21, 32, 33 (February 1981).
 - 234 Yates, Jean L. "Unix and the Standardization of Small Computer Systems." *Byte*, 160 – 166 (October 1983).
 - 235 Zachmann, Mark S. "The MWC-86C Compiler." *PC Magazine*, 124 – 129 (March 20, 1984).
 - 236 Zachmann, Mark S. "The Whitesmith's C Native Compiler." *PC Magazine*, 130 – 137 (March 20, 1984).
 - 237 Zintz, Walter. "A Survey of Unix and C Resources." *Byte*, 212 – 213 (August 1983).

Topical Index to C Bibliography

Advanced topics

004, 009, 011, 016, 017, 020, 027, 029, 047, 058, 073, 074, 120, 121, 123, 128, 129, 146, 161, 192, 193, 195, 199, 215, 216, 230

Algorithms

001, 002, 003, 072, 084, 085, 092, 232

BCPL

028, 123

Benchmarks

072, 079, 080, 081, 082, 084, 085, 110

Bibliographies

237

Book reviews

006, 008, 031, 123, 124, 156, 169, 188, 194, 222, 237

Now With
Flipscreen

SYMD

SYMBOLIC • DEBUGGER

Faster, easier more productive

SYMD is a unique new programming aid that dramatically reduces the time and effort needed to identify and correct programming errors. SYMD simplifies debugging by utilizing source code symbols and line numbers wherever an address or offset is required. Flipscreen

feature allows separate display, including graphics, for SYMD and program under test. SYMD also lets you: Examine and/or change real number data; assign permanent or temporary breakpoints, including pass counts to control looping; check instruction execution sequences with backtrace commands; use profiling commands to identify the most active

parts of a program. And much more. SYMD can be used with compiler or assembly language programs. System requirements: PC-DOS™ or MS-DOS™ 1.1 or 2.0; 192K recommended; 80-column display.

SYMD is priced at only \$125. Free literature on request. VISA or MasterCard accepted.

™MS-DOS is a trademark of Microsoft Corporation.

™PC-DOS is a trademark of the IBM Corporation.

D+V Systems

400 Amherst Street Nashua, NH 03063 (603) 881-7140

Circle no. 23 on reader service card.

Eco-C Compiler

Release 3.0

We think Rel. 3.0 of the Eco-C Compiler is the fastest full C available for the Z80 environment. Consider the evidence:

Benchmarks*

(Seconds)

Benchmark	Eco-C	Aztec	Q/C
Seive	29	33	40
Fib	75	125	99
Deref	19	CNC	31
Matmult	42	115	N/A

*Times courtesy of Dr. David Clark
CNC - Could Not Compile
N/A - Does not support floating point

We've also expanded the library (120 functions), the user's manual and compile-time switches (including multiple non-fatal error messages). The price is still \$250.00 and includes Microsoft's MACRO 80. As an option, we will supply Eco-C with the SLR Systems assembler - linker - librarian for \$295.00 (up to six times faster than MACRO 80).

For additional information,
call or write:



(317) 255-6476
6413 N. College Ave. • Indianapolis, Indiana 46220



NEW RELEASE

Circle no. 24 on reader service card.

PROGRAMMER'S DEVELOPMENT TOOLS

IBM Personal Computer Language and Utility Specialists

LANGUAGES:

	List	Ours
Lattice C Compiler	\$500	295
STSC APL*Plus/PC	Sale Priced	595 469
DeSmet C Compiler with Debugger	159	145
CB-86 by DRI	600	429
Instant-C by Rational Systems,		
Interpretive C	500	469
8088 Assembler w/Z-80 Translator		
2500 AD	100	89
C Programming System by Mark Williams	500	459

Call for Prices and Information about other Languages.

Special Holiday Season Sale Price! Computer Innovations C-86 Compiler \$278

Performance, Features and Low Price, make the C.I. C-86 a Holiday Season Value. Save over \$30 from our normal price of \$309. Call for more information and details.

UTILITIES:

****C Functions Library Sale****

C Utility Library for C-86
and Lattice \$149 119
New from Essential Software
Written 99% in C

The Greenleaf Functions for
C-86, Lattice, and Mark
Williams C Compilers 175 139

Each product features a full library of over
200+ C Functions. No Royalties. Both
include source code.

Communications Library

by Greenleaf	New	160	139
Btrieve by SoftCraft		245	199
C-Food Smorgasbord		150	109
Trace-86 by Morgan Computing		125	115
OPT-TECH Sort High Performance Utility		99	87
C Power Paks from Software Horizons	CALL	CALL	
Phact by Phact Associates		250	199
Plink-86 Overlay Linkage Editor		395	310
Panel Screen Design/Editing by Roundhill		350	234
Profiler by DWB & Associates		125	99
Halo Color Graphics for Lattice, C1-86		200	125
Graphic from Scientific Endeavors		195	179
Windows For C by Creative Solutions		150	109

A SOLID GOLD VALUE

CodeSmith-86 Debugger
Version 1.8 by Visual Age

Retail \$145. Our Normal Price \$129

Special Sale Price! \$109

Sale Price effective until 11/23/84.

Prices are subject to change without notice.

Call for our New Catalog consisting of
200+ Programmer's Development Tools
Exclusively for IBM PC's and Compatibles.



Visa/MC -
NO EXTRA CHARGE

Account is charged when order is shipped.

1-800-336-1166



Programmer's Connection
281 Martine Drive
Kent, Ohio 44240
(216) 678-4301 (In Ohio)

"Programmers Serving Programmers"

Circle no. 53 on reader service card.

- C-like languages
 - C.Plus
 - 076, 077, 078
 - tiny-C
 - 061, 083, 092, 110, 130
- C compiler reviews (by environment)
 - CP/M
 - 031, 066, 070, 079, 080, 081, 082, 090, 111, 130, 132, 137, 165, 167, 176, 182, 188
 - CP/M-86
 - 031, 106, 111
 - MSDOS (PCDOS)
 - 014, 015, 044, 104, 106, 111, 112, 168, 228, 235, 236
- C compilers (by name)
 - Microsoft C
 - 044, 104, 106, 111, 168
 - LC (Tandy/Radio Shack)
 - 052, 095
 - Supersoft C
 - 070, 079, 080, 081, 088, 106, 107, 111, 137, 168
 - Aztec C II
 - 082, 111, 112, 137
 - Software Toolworks C/80
 - 082, 094, 137
 - Intellect Associates C88
 - 104, 106, 111, 168
 - C-Systems
 - 104, 106, 111, 168, 228, 229
 - Computer Innovations Ci-C86
 - 104, 106, 111, 112, 168, 228
 - Telecon
 - 079, 080, 104, 111, 137, 168
 - Lattice
 - 014, 044, 104, 106, 111, 112, 168
 - DeSmet
 - 079, 080, 081, 104, 106, 111, 137, 168
 - Digital Research
 - 079, 080, 081, 104, 106, 111, 137, 168
 - Mark Williams
 - 079, 080, 081, 104, 106, 111, 137, 168, 235
 - B D Software C
 - 066, 079, 080, 081, 094, 104, 106, 111, 132, 137, 165, 167, 168, 174, 176, 208
 - Whitesmith's
 - 079, 080, 081, 104, 106, 111, 137, 168, 182, 236
 - tiny-C
 - 061, 079, 080, 081, 083, 092, 104, 106, 110, 111, 130, 137, 168
- C vs. Pascal comparisons
 - 065, 087, 110, 122, 139, 155, 171, 210
- Introductory tutorials
 - 013, 015, 031, 054, 079, 080, 081, 083, 091, 116, 117, 122, 125, 126, 140, 152, 186, 188, 211, 228
- Language definition
 - 031, 073, 074, 120, 122, 161, 185, 186, 187, 199
- Language extensions
 - 004, 005, 011, 016, 017, 027, 029, 030, 128, 129, 146, 207, 209, 215, 216, 230
- Language selection criteria
 - 005, 031, 047, 055, 056, 065, 084, 085, 087, 110, 141, 152, 153, 184, 192, 195, 210
- Programming hints (generally brief articles or letters)
 - 038, 049, 057, 075, 096, 115, 147, 158, 158a, 170, 171, 173, 191, 204, 205, 206, 231
- Programs (include source code)
 - command shells
 - 097a, 098, 099, 119, 183, 200, 204
 - concordance generators
 - 213
 - CP/M interfacing
 - 024, 067, 090, 137, 148
 - cross assemblers
 - 046
 - cross reference generators
 - 166, 219, 220
 - debuggers
 - 010, 016, 017, 027, 145
 - disk utilities
 - 050, 143, 144, 162, 163
 - editor (Ed Ream - ED2)
 - 042, 108, 177, 178
 - editor (Ed Ream - RED)
 - 179, 180
 - filters
 - 131, 191, 202, 218
 - Forth-like language
 - 021
 - help facilities
 - 214
 - hardware interfacing
 - 009, 024, 025, 047, 055, 056, 058, 105, 141, 189, 193, 212
 - operating systems
 - 097a, 098, 099, 159, 183
 - scientific instrumentation
 - 009, 024, 025, 047, 055, 056, 058, 105, 141, 189, 193, 212
 - Small-C (Cain)
 - 012, 019, 023, 032, 033, 048, 061, 086, 089, 093, 098, 119, 151, 158, 190, 200, 225, 227, 233
 - Small-C, v. 2 (Hendrix)
 - 034, 097, 100, 101, 102, 103, 197, 198
 - Small-VOS
 - 097a, 099, 183
 - telecommunications
 - 045, 047, 063, 175
 - time/date functions
 - 025
- RATFOR
 - 036, 037, 196
- Software design and development
 - 005, 020, 083, 122, 125, 126, 142, 150, 184, 195, 207
- Software portability
 - 021, 118, 120, 121, 122, 184, 206, 224
- Software (non-compilers) reviews
 - command shells
 - 036, 064, 067, 114, 134, 135, 138, 157, 183, 196, 203
 - MINCE text editor
 - 041, 043, 059, 060, 133, 154, 164, 170, 221
 - SCRIBBLE text formatter

- 136
- other software
- 183
- ED editor
 - 042, 108, 177, 178
- RED editor
 - 179, 180
- Software tools
 - 026, 036, 126, 183, 196, 223
- Unix-related
 - 007, 018, 026, 053, 063, 068, 069, 071, 090, 118, 121, 127, 142, 148, 149, 196, 202, 204, 206, 224, 234
- Writing tools
 - 039, 226

Public Domain C Software

At the present time a significant amount of public domain software has been written in the C language. Presented below is a brief synopsis of what is currently available. Each disk is a standard 8-inch CP/M disk or approximately 256K of software. Further information concerning ordering, media charges, contents, and so on can be obtained from the addresses preceding the diskette descriptions.

C Users' Group

415 E. Euclid, McPherson, KS 67460, (316) 241-1065.
Also available from Elliam Associates, 24000 Bessemer Street, Woodland Hills, CA 91367, (818) 348-4278.

The C Users' Group provides some of the classic materials in C. The editor (ED2) by Ed Ream is available in both Small-C and BDS C versions. The Small-C compiler is also available.

Five utility diskettes (Utilities I through V) provide programs for text compression, concordances, keyword-in-context concordance, disk editor (direct disk patcher), WP (ROFF-type) text formatter, NRO (enhanced ROFF-type) text formatter, several disk doctoring and patching programs, programs for data compaction and expansion, XSDIR (extended CP/M directory program), entab and detab (insert and remove tabs) programs, telecommunications programs, and a Unix modem system.

A smaller triad of function diskettes (Functions I through III) provide programs for directed input/output facilities (i.e., pipelines), floating point and long integer extensions to BDS C, a trigonometric function library, and scope (simple full-screen text editor).

The group also has available several diskettes of games (Games I and II and an Adventure diskette) that provide a needed respite from "heavy-duty" C programming. The games available include everything from the original (Crowther/Woods) Adventure to Othello, Hunt the Wumpus, etc.

For a much more specialized and smaller clientele, cross-assemblers are available for the 6809 chip (the Radio Shack color computer), the 6800, and the 1802 microprocessor chips.

An extensive set of diskettes (13 volumes) is devoted to the "Software Tools" of Kernighan and Plauger with the primitives translated into C.

Language enthusiasts can find an implementation of a

Forth-like language on the PISTOL (Portably Implemented STack-Oriented Language) diskette.

CPMUG

The CP/M Users Group, 1651 Third Avenue, New York, NY 10028.

Also available from Elliam Associates, 24000 Bessemer Street, Woodland Hills, CA 91367, (818) 348-4278; and New York Amateur Computer Club, Inc., P. O. Box 106, Church Street Station, New York, NY 10008.

Three C diskettes are available from the CPMUG. One (volume 48) is a sampler of material in the BDS C language. A second (volume 53) contains the original Crowther and Woods Adventure game, and the last diskette (volume 85) contains a very sophisticated set of text compression programs.

SIG/M Users Group

available from Elliam Associates, 24000 Bessemer Street, Woodland Hills, CA 91367 (818) 348-4278; and New York Amateur Computer Club, Inc., P. O. Box 106, Church Street Station, New York, NY 10008.

Three C diskettes are available from the SIG/M Users Group. One (volume 60) contains a very sophisticated set of text compression programs. A second (volume 114) contains an expanded version of the PISTOL diskette from the C User's Group, and the last diskette (volume 118) contains a C-based implementation of an experimental object-oriented language XLISP.

Some Other Sources of C Software

Below are listed some of the places you may find relatively inexpensive C software, usually including source code.

Edward K. Ream
1850 Summit Avenue
Madison, WI 53705
(608) 231-2952

Ed Ream can supply diskettes for his enhanced RED editor with source code for Small-C, BDS C, Aztec C, and the Digital Research C compilers.

Western Wares
Box C
Norwood, CO 81423
(303) 327-4898

Western Wares provides two diskettes of C utilities including a disk sector editor/dumper program (VIEW diskette) and a set of file utility programs (C-PACK diskette).

Northwest Microsystem Design
P. O. Box 10853
Eugene, OR 97401
(503) 484-7129

A full-screen editor (CSE) is available from Northwest Microsystem Design.

Blaise Computing
2034 Blake Street

Berkeley, CA 94704

An extensive set of tools for the IBM PC is available from Blaise Computing. These tools, available in three packages (C Tools, C Tools II, and View Manager), include such things as strings, screen access, access to BIOS IBM/PC, general DOS interface, and screen support.

Greenleaf Software
2101 Hickory Drive
Carrollton, TX 75006

Over 200 functions for strings, graphics, etc. (IBM PC) are available on the Greenleaf Functions diskette.

Que Corporation
7960 Castleway Drive
Indianapolis, IN 46250

The book *The C Programmer's Library* has available a companion diskette including such topics as recursion, linked lists, sorting algorithms, Huffman text compression, etc.

Dedicated Micro Systems
P. O. Box 481
Chanute, KS 66720
(316) 431-0018

Two diskettes provide extended precision floating point extensions to BDS C. The BDS C compiler is also available.

J. E. Hendrix
Box 8378
University, MS 38677-8378

Mr. Hendrix can provide diskettes for his Small-VOS operating system, for his Small-C compiler, and for a set of software tools written in Small-C (small-tools).

Algorithmic Technology
P. O. Box 278
Exton, PA 19341-0278
(215) 363-7028

Algorithmic Technology publishes a massive directory (40+ pages) of public domain software for CP/M-80, PC DOS, MSDOS, BDS C, and Software Toolworks C/80.

Diskette Conversion Firms

Finally, due to the diversity of disk formats available, I have included below three firms who provide diskette format conversion facilities.

Elliam Associates, 24000 Bessemer Street, Woodland Hills, CA 91367, (818) 348-4278.

Fred Greeb, LogiCom, Inc., P. O. Box 27465, Lakewood, CO 80227, (303) 986-6651.

Mycroft Labs, 2369 North Monroe St., Box 68, Suite B-188, Tallahassee, FL 32303, (904) 385-1141.

[We understand, however, that Mycroft may be phasing out its conversion activities.—ED]

DDJ

Reader Ballot

Vote for your favorite feature/article.
Circle Reader Service No. 195.

BUFFERED I/O BOARD Introductory Price *\$59.95
With despool functions, protocols supported: XON/XOFF, ETX: ETB/ACK

80 CHARACTER VIDEO BOARD *\$49.50
25 Lines with status, compatible with Wordstar & dBase

Includes Bareboard, Heatsink & Documentation Call or write for more information.

SPO
Simpliway Products Co.
P.O. Box 601
Hoffman Estates, IL 60195
(312) 359-7337

OEM dealer pricing available, \$3.00 S/H, IL. Res. add 7% tax
dBase™ - of Ashton-Tate Corp. — Wordstar™ - of Micropro Int'l. Corp.

VISA
MasterCard

Circle no. 63 on reader service card.

Elegance
Power
Speed



C Users' Group
Supporting All C Users
Box 97
415 Euclid
McPherson, Kansas 67460
(316) 241-1065

Circle no. 17 on reader service card.

Put Dr. Dobb's Under the Christmas Tree!

Treat your friend (or yourself!) to Dr. Dobb's Journal

BOUND VOLUME 7

Every 1982 Issue Available For Your
Personal Reference.

Vol. 1 1976

The material brought together in this volume chronicles the development in 1976 of Tiny BASIC as an alternative to the "finger blistering," front-panel, machine-language programming which was then the only way to do things. This is always pertinent for bit crunching and byte saving, language design theory, home-brew computer construction and the technical history of personal computing.

Topics include: Tiny BASIC, the (very) first word on CP/M, Speech Synthesis, Floating Point Routines, Timer Routines, Building an IMSAI, and more.

Vol. 2 1977

1977 found DDJ still on the forefront. These issues offer refinements of Tiny BASIC, plus then state-of-the-art utilities, the advent of PILOT for microcomputers and a great deal of material centering around the Intel 8080, including a complete operating system. Products just becoming available for reviews were the H-8, KIM-1, MITS BASIC, Poly Basic, and NIBL.

Articles are about Lawrence Livermore Lab's BASIC, Alpha-Micro, String Handling, Cyphers, High Speed Interaction, I/O, Tiny Pilot & Turtle Graphics, many utilities, and even more.

Vol. 3 1978

The microcomputer industry entered its adolescence in 1978. This volume brings together the issues which began dealing with the 6502, with mass-market machines and languages to match. The authors began speaking more in terms of technique, rather than of specific implementations; because of this, they were able to continue laying the groundwork industry would follow. These articles relate very closely to what is generally available today. Languages covered in depth were SAM76, Pilot, Pascal, and Lisp, in addition to RAM Testers, S-100 Bus Standard Proposal, Disassemblers, Editors, and much, much more.

Vol. 4 1979

This volume heralds a wider interest in telecommunications, in algorithms, and in faster, more powerful utilities and languages. Innovation is still present in every page, and more attention is paid to the best ways to use the processors which have proven longevity—primarily the 8080/Z80, 6502, and 6800. The subject matter is invaluable both as a learning tool and as a frequent source of reference.

Main subjects include: Programming Problems/Solutions, Pascal, Information Network Proposal, Floating Point Arithmetic, 8-bit to 16-bit Conversion, Pseudo-random Sequences, and Interfacing a Micro to a Mainframe—more than ever!

Vol. 5 1980

All the ground-breaking issues from 1980 in one volume! Systems software reached a new level with the advent of CP/M, chronicled herein by Gary Kildall and others (DDJ's all-CP/M issue sold out within weeks of publication). Software portability became a topic of greater import, and DDJ published Ron Cain's immediately famous Small-C compiler—reprinted here in full!

Contents include: The Evolution of CP/M, a CP/M-Flavored C Interpreter, Ron Cain's C Compiler for the 8080, Further with Tiny BASIC, a Syntax-Oriented Compiler Writing Language, CP/M to UCSD Pascal File Conversion, Run-time Library for the Small-C Compiler and, as always, even more!

Vol. 6 1981

1981 saw our first all-FORTH issue (now sold out), along with continuing coverage of CP/M, small-C, telecommunications, and new languages. Dave Cortesi opened "Dr. Dobb's Clinic" in 1981, beginning one of the magazine's most popular features.

Highlights: information on PCNET, the Conference Tree, and The Electric Phone Book, writing your own compiler, a systems programming language, and Tiny BASIC for the 6809.

Vol. 7 1982

In 1982 we introduced several significant pieces of software, including the RED text editor and the Runic extensible compiler, and we continued to publish utility programs and useful algorithms. Two new columns, The CP/M Exchange and The 16-Bit Software Toolbox, were launched, and we devoted special issues to FORTH and telecommunications. Resident Intern Dave Cortesi supplied a year of "Clinic" columns while delivering his famous review of JRT Pascal and writing the first serious technical comparison of CP/M-86 and MSDOS. This was also the year we began looking forward to today's generation of microprocessors and operating systems, publishing software for the Motorola 68000 and the Zilog Z8000 as well as Unix code. And in December, we looked beyond, in the provocative essay, "Fifth-generation Computers."

YES! ☐ Please send me the following Volumes of **Dr. Dobb's Journal**.
☐ ALL 7 for ONLY \$165, a savings of over 15%!

Payment must accompany your order.

Please charge my: ☐ Visa ☐ MasterCard ☐ American Express

I enclose ☐ Check/money order

Card # _____ Expiration Date _____

Signature _____

Name _____ Address _____

City _____ State _____ Zip _____

Mail to: Dr. Dobb's Journal, 2464 Embarcadero Way, Palo Alto, CA 94303

3023 Allow 6-9 weeks for delivery. Circle no. 83 on reader service card.

This offer expires February 28, 1985

Vol. 1	_____ x	\$26.75 =	_____
Vol. 2	_____ x	\$27.75 =	_____
Vol. 3	_____ x	\$27.75 =	_____
Vol. 4	_____ x	\$27.75 =	_____
Vol. 5	_____ x	\$27.75 =	_____
Vol. 6	_____ x	\$27.75 =	_____
Vol. 7	_____ x	\$30.75 =	_____
All 7	_____ x	\$165.00 =	_____

Sub-total \$ _____

Postage & Handling _____
Must be included with order.

Please add \$1.25 per book in U.S.
(\$2.00 each outside U.S.)

TOTAL \$ _____

RESORT

by Donald G. Krantz

One of the things I spend a lot of time doing is writing. I write manuals, a column for the local club newsletter, letters, worksheets, and tests for my classes at night school. Aside from my copy of WordStar, the program I use the most is my public domain spelling checker SPELLM11 by Michael C. Adler.*

While I like to write, I am not what you would call a superlative speller. Until I got SPELLM11, I thought "nes-sesary" [sic] was just fine. In fact, I almost added it to SPELLM11's user dictionary.

One of the chores associated with any spelling checker is the maintenance of the user dictionary. Because much of what I write is technobabble, the checker flags many of the words that I use as unrecognized. The user dictionary to SPELLM11 must be in alphabetical order.

Being a basically lazy person, I wrote the attached program (see the listing on page 90) to read through a document after spelling corrections, strip the speller's flag (a leading tilde ~), and add

correct words unknown to the master dictionary DICT.DIC to the user dictionary SPELL.DIC.

I used the sorting technique from *The C Programming Language* by Kernighan and Ritchie (Prentice-Hall, 1978): the lowly binary tree. This is my favorite sorter, as it is easily implemented in C.

The program takes the input document filename from the command line, opens it for input, and creates a temporary file (with extension .\$\$\$) for output. The input file is scanned for tildes, which flag unrecognized words. When a tilde is found, the word following is converted to upper case and added to a binary tree.

As the input file is scanned, it is fed to the temporary file, minus the tildes, with any high-order bits (used in some editors for formatting purposes) intact. When the input file is exhausted, it is renamed with extension .BAK, and the temporary file takes the original input filename.

The user dictionary SPELL.DIC is then opened, read in, and sorted into

***"RESORT is a utility which maintains a user dictionary
SPELL.DIC in alpha order."***

Donald G. Krantz, 2845 42nd Avenue South, Minneapolis, MN 55406.

* *SPELLM11* by Michael C. Adler (public domain) is available as Volume 1 from Avos Blind User's Group, 1485 Energy Park Drive, St. Paul, MN 55108—\$5 for 5¼-inch (specify format) and \$10 for 8-inch ss/sd CP/M—or from AVOS/TCOG BBS, (612) 646-2848, 300/1200 baud, 24 hours, XMODEM protocol.

the tree. Finally, the tree, sorted into alphabetical order, is written back to the user dictionary.

For those unfamiliar with the binary tree, it is important that the files be read in this order. If a sorted array is loaded into a binary tree, the tree becomes extremely unbalanced, degenerating into a linear linked list; the program bogs down, chasing up and down the tree. For best results, a binary tree should have new nodes presented randomly.

The code is fairly straightforward, as C code goes. It is as close to Unix version 7 compatible as I could make it without actually compiling it on Unix version 7 (my VAX is in the shop). A function-by-function breakdown follows.

main()

main() is concerned mostly with shuffling filenames around. It's one of those run-on functions; for that I apologize. The calls to **sort()**, **sort2()**, and **print()** do all the program's work.

makebak()

makebak() takes the input filespec from the command line and examines it to see if an extension is part of the filespec. If so, the extension is replaced by **\$\$\$** for the temporary filespec and **.BAK** for the backup filespec. If no extension is included in the original filespec, the appropriate extensions are appended to the backup and temporary filespecs.

sort()

sort() scans the input stream for tildes. When it finds one, it accumulates subsequent alphabetic characters into the array **w[]**. The words then are passed to **treeload()**, which assembles the binary tree.

sort2()

sort2() scans the user dictionary using basically the same tokenizing routine as **sort()**, except that it accumulates all of the words from the input stream, not just words flagged by tildes. The words again are passed to **treeload()**. This routine is the speed bottleneck in the program. Because the user dictionary is already in order, the more words added to the tree from the input document (presumably in random order, which tends to start the tree out in a balanced condition), the faster this phase will run.

treeload()

This is patterned directly after Kernighan and Ritchie (K & R). (My philosophy of life is that no matter how many times you invent a wheel, a good one will still be round.) This function chases down the binary tree until it finds either a duplicate word or the end of the tree. Should it find the end of the tree, it creates a new node.

print()

print() writes the new user dictionary back to disk. This routine illustrates why I like binary trees so much:

you can write such elegant code. Again, it's from K & R. **print()** writes all daughter nodes lower than a given node, then the node itself, and then all daughter nodes higher than itself. Try to do this in BASIC sometime.

getc()

getc() transfers one character from the input file to the temporary file and returns the same character to the tokenizer. Note that ints are used to allow **ERROR** to be passed back to the caller. If chars were used, **ERROR** (usually **-1**, or **0xFFFF**) would be converted to **DELETE** (**0xFF**).

Ian Ashdown (a referee for *Dr. Dobb's*) suggested that the use of the function **sort2()** is a misapplication of the binary tree because of the previously noted tendency towards unbalancing the tree by adding words in order. He favors using a routine (his pseudocode appears in the figure below) to merge the presumably alpha-

betic word stream from the user dictionary file with an alphabetic stream taken from the binary tree. This method certainly would eliminate the speed bottleneck noted in **sort2()**, but it has one drawback: if the user accidentally gets the dictionary file out of order by editing it (it is, after all, an ASCII file), this merge routine will never again produce a sorted output. I opted to trade some speed losses for reliability.

The program shell can be modified easily to do other tasks unrelated to spelling. For example, it would be a matter of adding a few lines of code to do word frequency counts. I have made a few changes to produce an index generator. In fact, I have written a tokenizer for a preprocessor that uses about half the code in this program. **DDJ**

(Listing begins on page 90)

Reader Ballot

Vote for your favorite feature/article.
Circle Reader Service No. **196**.

```

/* -----
                        merge( ) - merges two alphabetical streams.
----- */

merge( )
{
    get( a word "word_sd" from SPELL.DIC )
    get( a word "word_bt" from binary tree )
    do
    {
        if( "word_sd" <= "word_bt" ) /* compare words */
        {
            put( "word_sd" in output stream )
            get( another word "word_sd" from SPELL.DIC )
        }
        else
        {
            put( "word_bt" in output stream )
            get( another word "word_bt" from binary tree )
        }
    }
    write( any remaining words from SPELL.DIC or binary tree )
    /* only one of the two can have leftover words */
}

```

Figure
Ian Ashdown's Pseudocode for Merge

Resort Listing

(Text begins on page 88)

```
/*-----
RESORT is a utility which maintains a user dictionary SPELL.DIC
in alpha order. This works in conjunction with the utility
SPELLM11.COM, by Michael C. Adler, as modified for the Osborne
User Groups.

RESORT is Copyright (C) 1984 by Donald G. Krantz. No commercial
use may be made of this program without permission of Author.

Limited license is granted for non-commercial distribution of
this program in either source or object form.
-----*/

#define VERSION "RESORT Version 1.1 (C) 1984 Donald G. Krantz"
#include "stdio.h"

/*-----
structure node is the basic data type used to describe the
sorted array of words.
-----*/

struct node {
    struct node *lower;
    struct node *higher;
    char *word;
};

struct node *root;      /* root is the root of the binary tree */

FILE *inptr;            /* file descriptor to input files */
FILE *outptr;           /* file descriptor to output file */

main( argc, argv )
    int argc;
    char *argv[];
{
    char tempfile[ 20 ];    /* temporary text filespec */
    char bakfile[ 20 ];    /* .BAK filespec */

    printf("\n\n", VERSION );
    /* check command line argument */
    if( argc != 2 )        /* one argument only... */
    {
        printf("\nERROR - Use: \nRESORT d:filename.typ\n\n");
        exit( 1 );
    }
    /* open input file */
    if( (inptr = fopen( argv[1], "r", 512 )) == NULL )
    {
        printf("\nERROR - Can't open file \s\n",argv[1]);
        exit( 1 );
    }
    /* make .BAK and .$$$ file spec and open temps */
    makebak( argv[ 1 ], bakfile, tempfile );
    unlink( tempfile );    /* dump old temps */
    if( (outptr = fopen( tempfile, "w" ) ) == NULL )
    {
        printf( "\nERROR - Can't open temporary file\n" );
        exit( 1 );
    }

    /* sort input file and strip tildes, transfer to temp file */
    root = sort( NULL );
    /* change temp filename to original, original to .BAK */
    fclose( inptr );        /* close input */
    if( fclose( outptr ) == ERROR )    /* close temp */
    {

```



```

        printf( "\nERROR - Disk full - aborting\n" );
        unlink( tempfile );
        exit( 1 );
    }
    unlink( bakfile );                                /* dump old .BAK */
/* rename( new_file_spec, old_file_spec ) */
    rename( bakfile, argv[1] );                        /* make new .BAK */
    rename( argv[1], tempfile );                       /* make new text */
    if( root == NULL )                                /* no new words */
        exit( 0 );                                    /* so quit */
/* sort .DIC file into binary tree */
    if( (inptr = fopen( "SPELL.DIC", "r" )) == NULL )
    {
        printf( "\nERROR - Can't open dictionary file\n" );
        exit( 1 );
    }
    root = sort2( root );                            /* sort .DIC */
    fclose( inptr );                                  /* dump .DIC */
    /* re-write .DIC file */
    if( (inptr = fopen( "SPELL.DIC", "w", 512 )) == NULL )
    {
        printf( "\nERROR - Can't write to dictionary file\n" );
        exit( 1 );
    }
/* write sorted tree to .DIC file */
    print( root );
    if( fclose( inptr ) == ERROR )
    {
        printf( "\nERROR - Disk full - Recheck SPELL.DIC\n" );
        exit( 1 );
    }
}

```

```

/*-----
makebak() creates filespecs for a temporary file and a backup file
to be used in filtering applications like RESORT. It's kind of
a generic function.
-----*/

```

```

makebak( orgfile, bakfile, tempfile )
char *orgfile, *bakfile, *tempfile;
{
    strcpy( tempfile, orgfile );
    strcpy( bakfile, orgfile );
    if( index( tempfile, '.' ) == NULL )    /* .TYP spec'ed? */
    {                                       /* no */
        strcat( tempfile, ".$$$" );
        strcat( bakfile, ".BAK" );
    }
    else                                   /* yes */
    {
        strcpy( index( tempfile, '.' ), ".$$$" );
        strcpy( index( bakfile, '.' ), ".BAK" );
    }
}

```

```

/*-----
sort() reads the input file and puts the words with leading
tildes into the binary tree.
-----*/

```

```

struct node *
sort( p )
{
    struct node *p;

    static char w[100];
    register int i;
    static int c;

    while( TRUE )
    {

```

(Continued on next page)

Resort Listing (Listing Continued, text begins on page 88)

```
/* find tilde */
while(((c = getcc( inptr )) & 0x7F) != '~') &&
      (c != ERROR))
    ;
/* accumulate subsequent word into 'w' */
i = 0; /* i = letter count in word */
while( (i == 0) || (isalpha( c ) ) )
{
    c = getcc( inptr );
    if( c == ERROR )
        return( p );
    c = toupper( 0x7F & c );
    if( isalpha( c ) == FALSE )
        break;
    w[ i++ ] = c;
}
if( i == 0 )
    continue;
w[ i ] = '\0';
/* put word into binary tree */
p = treeload( w, p );
}

/*-----
sort2() reads the .DIC file and puts the words into the binary
tree.
-----*/

struct node *
sort2( p )
    struct node *p;
{
    static char w[100];
    register int i;
    static int c;

    while( TRUE )
    {
        i = 0; /* i=letter count in word */
        /* loop for alphabetic data, skip non-alpha */
        while((i == 0) || (isalpha( c )))
        {
            c = getc( inptr );
            if( c == ERROR )
                return( p );
            c = toupper( 0x7F & c );
            if( isalpha( c ) == FALSE )
                break;
            w[ i++ ] = c;
        }
        if( i == 0 )
            continue;
        w[ i ] = '\0';
        /* put word into binary tree */
        p = treeload( w, p );
    }
}

/*-----
treeload( w, p ) loads the tree with words, placing them in
alphabetical order.
-----*/
```



```

struct node *
treeload( w, p )
    char *w;
    struct node *p;
{
    if( p == NULL )
/* we are at end of tree, need to add a node for this word */
    {
        if((p = alloc( sizeof( struct node ) )) == NULL)
        {
            printf("\nERROR - Out of memory");
            exit( 1 );
        }
        if((p->word = alloc( strlen( w ) + 1 )) == NULL)
        {
            printf("\nERROR - Out of memory");
            exit( 1 );
        }
        p->lower = NULL;
        p->higher = NULL;
        strcpy( p->word, w );
        return( p );
    }
    else if( strcmp( p->word, w ) > 0 )
/* word 'w' is lower than the word at this node, look lower */
        p->lower = treeload( w, p->lower );
    else if( strcmp( p->word, w ) < 0 )
/* word 'w' is higher than the word at this node, look higher */
        p->higher = treeload( w, p->higher );
/* if word 'w' is identical to the word at this node, we do */
/* nothing. In all cases, we return the current node pointer. */
    return( p );
}

```

```

/*-----
print() writes the words in alpha order back into the .DIC
file.
-----*/

```

```

print( p )
    struct node *p;
{
/* print all words lower than current node first */
    if( p->lower != NULL )
        print( p->lower );

/* print the word at the current node */
    fprintf( inptr, "\s\n", p->word );

/* print all words higher than the current node */
    if( p->higher != NULL )
        print( p->higher );
}

```

```

/*-----
getcc() gets chars from the infile, and writes all but tildes
to the outfile. This function does not disturb any high bits
set for parity or formatting.
-----*/

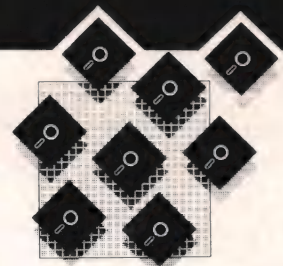
```

```

getcc( fd )
    int *fd;
{
    int c;
    c = getc( fd );
    if(((c & 0x7F) != '~') && (c != ERROR))
        fputc( c, outptr );
    return( c );
}

```

End Listing



Windows for C, Version 2.00

Company: Creative Solutions, 21
Elm Avenue, Richford, VT
05476

Computer: IBM PC and MSDOS (var-
ious C compilers)

Price: \$150.00, demo disk and
manual: \$30.00

Circle Reader Service No. 127

Reviewed by Ian Ashdown

We have good news: If you are running MSDOS on your IBM PC and programming C, you probably will be pleased to learn that Windows for C can offer you a convenient and reliable means of implementing windows for your text-handling programs.

Windows are, of course, those rectangles on a terminal's screen that enclose information not directly related to other information shown. In essence, they are screens in their own right, showing anything from a different part of the same text file to the operation of a completely independent, concurrently running program. Properly implemented, they can greatly enhance the utility of everything from menus and help facilities to full-screen editors, spreadsheets, data base managers, and multitasking operating systems.

As such, it behooves the C programmer to add windowing to his/her library of C functions. Windows for C greatly eases this effort by providing over 30 such functions for the IBM PC with MSDOS. Using this software, you can define and manipulate windows for text applications, treating each window as a separate screen and having the number of screens limited only by the available memory. If you don't need to display a particular window at any time, simply erase it from the screen. Because it is stored in memory, complete with current contents, it can be retrieved later and displayed.

In defining a window, you can speci-

fy its location and size and also define the visual appearance of its border and background. These characteristics are not static—you can dynamically redefine them at any time during program operation. Windows can also be overlapped, nested, cleared, and removed from the screen as required, all without affecting any other nonoverlapping windows.

The contents of the windows are whatever textual information you want; Windows for C supports all of the text mode capabilities of the IBM monochrome and graphics display adapters. Video attributes such as intensity, underline, reverse, blink, and color can be controlled on a character-by-character basis. The text can be written to the window with or without word wrap, and both vertical and horizontal scrolling are supported. If desired, you can also have automatic scrolling of text in a full window when the cursor reaches the window's border.

Each window has its own virtual cursor, with the cursor position always kept in memory. Accessing the window automatically moves the real screen cursor to the position of the virtual cursor, and any manipulation of the screen cursor in the window automatically updates the position of the virtual cursor in memory.

Text can come from any source—the terminal (or other input device) or a disk file. Once displayed in a window, text can be manipulated and, if desired, transferred to other windows. No general-purpose text editing functions are provided with Windows for C, but source code is provided for the routines to read the disk files into memory and into a window. Using this basic code as an example, the well-seasoned C programmer can write the interfaces necessary to link the functions provided by Windows for C to other application programs.

Windows for C displays and manipulates text by defining every window as a data structure, with variables for window position and size, virtual cursor position, background, text margins, word wrap and scroll switches, and so on. Also included in the window structure are pointers to the associated file and border structures. The file structure contains the name of the device or file being accessed and pointers to the part of the file currently displayed in the window, while the border structure defines the characters that form the horizontal and vertical sides and the corners of the window border.

The functions provided by Windows for C were written in both C and assembly language, and so far we have not found any bugs in them. The video output is fast and clean, showing no snow with either the graphics or monochrome display. (Some skewing is evident when the text is scrolled sideways, but this is not objectionable.) The text is sent directly to the video display buffer using line-oriented assembly language subroutines; word wrap, attribute selection, and automatic scrolling are user selectable.

A software package must be judged in large part by its documentation; obscure language, incorrectly described program actions, and undocumented features are the basis of many a programmer's nightmares. Happily, Windows for C is documented in well-written and readable English, and it appears to be all there. The manual we received consisted of 53 double-sided 8½ × 11-inch pages in a three-ring binder. The main topic being the window functions, each one was clearly and concisely documented in the style of Bell Laboratories' *Unix Programmer's Manual*, with entries for the function's name, usage, operation, definition, what it returns, and cautionary notes. For a reference manual, this style of

writing is ideal.

Moreover, in addition to the perfunctory "Getting Started," "Tutorial," and "Advanced Capabilities" sections, the manual has an appendix entitled "Mnemonic Abbreviations." This is a most welcome feature. How often have you gone in search of the meaning of an abbreviation such as "csr"? Windows for C lists them all in one place ("csr" refers to the screen cursor). Many other software documentation writers, take note!

To summarize the good news, we liked Windows for C: it is a very nice software package that has obviously been well thought out and very cleanly executed.

... and with the good news, a bit of the not-so-good. As previously noted, Windows for C was written for the IBM PC. It can also be used with unspecified "video-compatible" computers running MSDOS. However, since the manual does not say what IBM work-alikes are video compatible, you would do well to enquire before purchasing the package if you plan on using anything other than an IBM PC.

Most of the functions included in Windows for C are in object code form. Since each C compiler uses a different library format, Creative Solutions is obliged to distribute a different version of Windows for C for each one. C compilers currently supported are those distributed by Lattice, Microsoft, Computer Innovations (C86), and DeSmet.

You might also bear in mind that windows are very much in demand for personal computer software these days. Considering Microsoft's Windows, Digital Research's windowing version of Concurrent CP/M, and the host of similar software packages soon to follow, we must ask whether a software package that provides windowing functions for text modes only is desirable or sufficient. Mind you, these others are really operating system environments, whereas Windows for C enables you to develop stand-alone software to run under MSDOS.

The above notwithstanding, we can say that Windows for C is probably a worthwhile investment for the serious C programmer. Under the terms of Creative Solution's program license agreement, the functions provided can be

linked into the programmer's own software efforts and marketed without royalties or Creative Solution's copyright notice. As such, Windows for C is a professionally oriented set of programming tools and one that we can recommend.

Special thanks is extended to Robert Koorbatoff, free-lance programmer, who assisted in this review by providing his time, expertise, and Microsoft C compiler.

**MYCHESS, MSDOS
Version 1.0**

**Company: The Software Tools
works, 15233 Ventura Blvd.,
Suite 1118, Sherman Oaks, CA
91403**

**Computer: IBM PC, PC XT, or PCjr
Price: \$34.95**

Circle Reader Service No. 129

Reviewed by Ronald G. Parsons

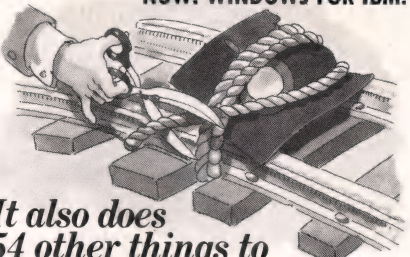
MYCHESS is a chess playing program for the IBM PC. Other versions are also available for CP/M. On the IBM PC with a graphics card, the display shows very attractive representations of the various chess pieces in high-resolution black and white. The system is configurable to use medium-resolution graphics or an ASCII representation on a monochrome display, but the attractiveness and readability are much degraded. The authors claim a U.S. Chess Federation rating of 1568.

You can set the skill level of the program by altering the number of half moves (ply depth) that the program looks ahead. It can play black or white, from the beginning or from a setup position, or you can interrupt a game, save it, and resume it later. The program can even play itself. The moves are recorded on the screen (showing the last eight moves), or the moves can be recorded on a printer or listed on a file at the end of the game for later viewing.

You start the game by answering a series of questions regarding the game setup. Then the chess board is displayed, and "white" plays. The notation identifies each square by a file label (column) A - H and a row label 1 - 8; for example, the white king starts on E1. Each move is denoted by a starting square and a destination; e.g., white king's pawn to king four would be E2 - E4. Illegal moves are rejected with a beep. Pawns can be pro-

**Of course,
POWER!™ saves
your Bad Disk.**

NOW! WINDOWS FOR IBM!



*It also does
54 other things to
keep your disk in line.*

**EVERYTHING YOU ALWAYS
WANTED
TO DO, BUT WERE AFRAID TO TRY**

Unlike some utility programs that are a headache to use, POWER! is engineered to spoil you with 55 features, simple and uniform commands, and utter simplicity of use. POWER! automatically alphabetizes and numbers your files. You select by the number and never type file names again. Need to [COPY], [RENAME], [ERASE], or [RUN] programs? Just type in their menu number! POWER! also locks out your disk's bad sectors [TEST] without destroying files—a critical difference from other utilities that search and destroy, without informing you what they've done, leaving you to wonder why your programs won't run. (And POWER! still has 50 commands to go!)

POWER! ONE PROGRAM DOES IT ALL!

You may own a few utility programs for your computer housekeeping, each with its own commands to memorize. POWER! has all the programs rolled into one 16K integrated package, so you do things you've never tried before—every day. Save sensitive data from prying eyes with [PASS] word protect, move a block of memory [MOVE], look for data [SEARCH] or compare files [CHECK]. POWER! also makes easy work of patching, [DISPLAY/SUBSTITUTE], customizing software [LOAD/SAVE]. Among the other commands are [SIZE], [STAT] [LOG], [DUMP], [TYPE], [JUMP], [FILL], [SET], and the CP/M version lets you restore erased files—even when you don't remember the filename—at a flick of the POWER! [RECLAIM] command. (Still 31 commands to go!)

**POWER! NOW FOR IBM's PC-DOS
AS WELL AS CP/M**

We first developed POWER! for CP/M two years ago, and a stack of testimonials from FORD to XEROX testify to its excellence. For IBM-PC™ users, special features like managing sub-directories, [CHANGE], and a separate creation of up to 8 simultaneous, on-screen [WINDOWS] have been added.

**MONEY-BACK GUARANTEE AND
A 10 DAY TRIAL**

POWER! has the Seal of Approval from the Professional Software Programmers Association, and you, too, must be happy with POWER!—or your money back! For only \$169 you can now really be in control of your computer. Call Computing! at (415) 567-1634, or your local dealer. For IBM-PC or any CP/M machine. Please specify disk format.

The company that earns its exclamation point.

COMPUTING!

2519H Greenwich, San Francisco, CA 94123

**TO ORDER CALL 800 TOLLFREE
800-428-7825 Extension 96H
In CA: 800-428-7824 Extension 96H**

IBM and IBM-PC are registered trademarks of
International Business Machines Corporation.

moted, but only one queen of each color may exist at any time.

The program is easy to play and learn. The moved piece flashes several times in its new position so that an eye blink does not cause you to miss a move. The move is also recorded on the right of the screen. I would like, however, to see several features added to the program. There is no way to tell whether you forgot to hit the enter key or the computer is thinking—a visible cursor would be helpful. There is no list of pieces captured—a command to list the captured pieces would be helpful. An on-line help feature to refresh your memory of the available commands also would be nice. A mouse interface would at least be cute.

The major impression of the type of

game played by MYCHESS is one of attrition. In fact, the program looks at capture moves first, which reduces the number of positions it must analyze to determine the next move. The priorities of move generation used by MYCHESS verifies this. The top four priorities of possible moves are:

- (1) Best variation from a previous iteration
- (2) Winning or even captures (thus the attrition)
- (3) Castle moves
- (4) *En passant* captures

In addition, a one-ply search is followed by two-ply searches, and so on, so that if a move is forced, the best move of the previous iteration is available. Also each successive iteration is based on the best variation predicted

by the previous iteration. The program examines one extra ply before backing up from a best variation if the side to move has anything to capture.

I noted only one bug and it was minor. One start-up option is to list the best variations for a selection of moves. If the number of moves suggested is more than three or so, the suggested moves wrap around and cover the board part of the screen.

MYCHESS is easy to use and presents a pleasant interface to the user. Without a graphics board and display, the program is much more difficult to use. I would recommend it especially to beginning chess players.

DDJ

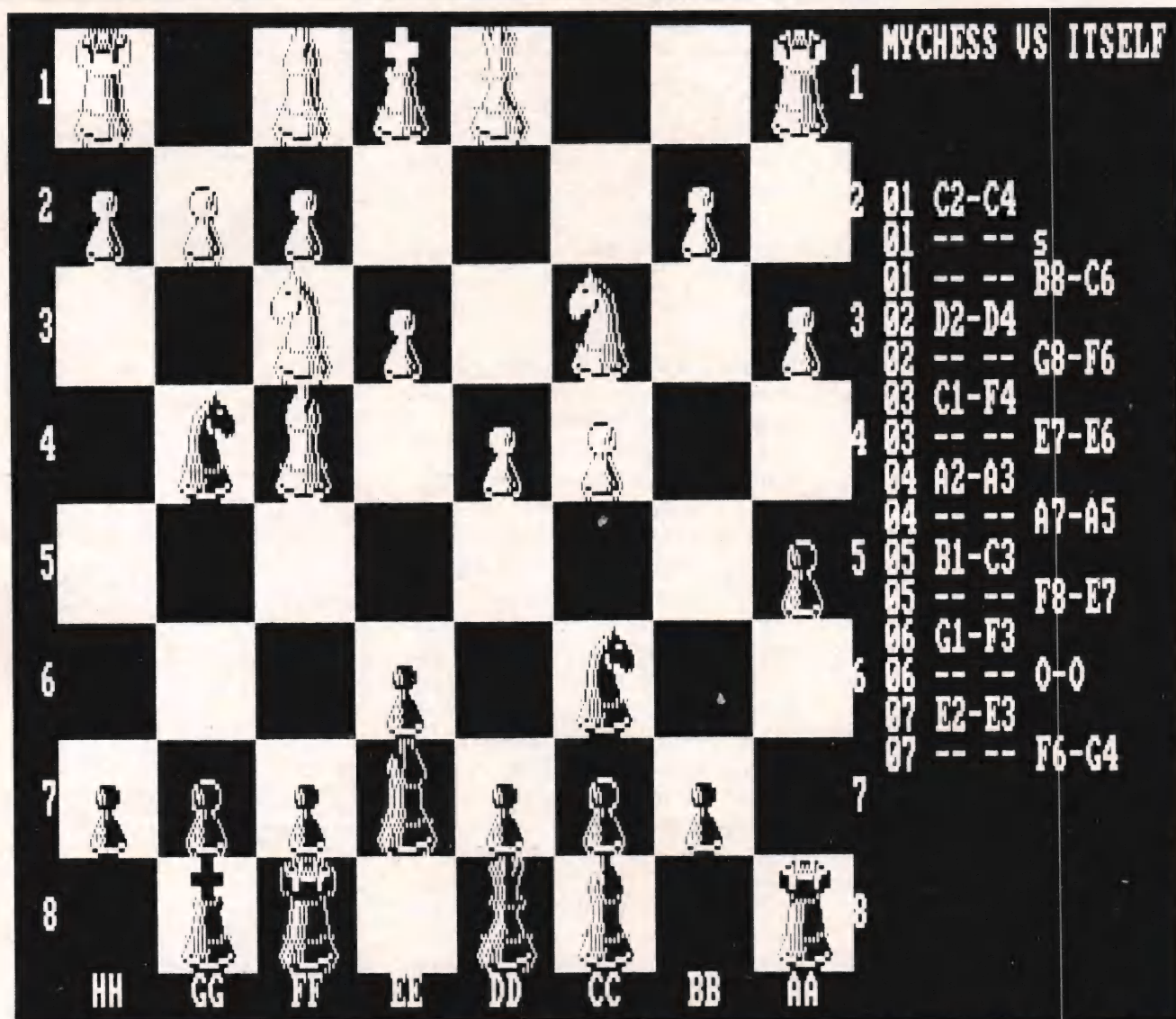


Figure 1
MYCHESS vs Itself

C Programmers: Program three times faster with *Instant-C™*

Instant-C™ is an optimizing **interpreter** for C that makes programming three or more times faster. It eliminates the time wasted by compilers. Many repetitive tasks are automated to make programming less tedious.

- Two seconds elapsed time from completion of editing to execution.
- **Symbolic debugging**; single step by statement.
- Compiled execution speed; 40 times faster than interpreted Basic.
- Full-screen editor integrated with compiler; compile errors set cursor to trouble spot.
- Directly generates .EXE or .CMD files.
- Follows K & R—works with existing programs. Comprehensive standard C library with source.
- Integrated package; nothing else needed.
- Works under PC-DOS*, MS-DOS*, CP/M-86*.

More productivity, less frustration, better programs. **Instant-C™** is \$500. Call or write for more info.

**Rational
Systems, Inc.**

(617) 653-6194
P.O. Box 480
Natick, Mass. 01760

Trademarks: MS-DOS (Microsoft Corp.), PC-DOS (IBM), CP/M-86 (Digital Research, Inc.), Instant-C (Rational Systems, Inc.)

Circle no. 58 on reader service card.

OPT-TECH SORT™

SORT/MERGE program for IBM-PC & XT

Now also sorts dBASE II files!

- Written in assembly language for **high performance**
Example: 4,000 records of 128 bytes sorted to give key & pointer file in 30 seconds. **COMPARE!**
- Sort ascending or descending on up to nine fields
- Ten input files may be sorted or merged at one time
- Handles variable and fixed length records
- Supports all common data types
- Filesize limited only by your disk space
- Dynamically allocates memory and work files
- Output file can be full records, keys or pointers
- Can be run from keyboard or as a batch command
- Can be called as a subroutine to many languages
- Easy to use, includes on-line help feature
- Full documentation — sized like your PC manuals
- **\$99** —VISA, M/C, Check, Money Order, COD, or PO
Quantity discounts and OEM licensing available

To order or to receive additional information
write or call:

OPT-TECH DATA PROCESSING

P.O. Box 2167 Humble, Texas 77347
(713) 454-7428

Requires DOS, 64K and One Disk Drive

Circle no. 49 on reader service card.

Super Fast

Get Fast Relief!

S-100! IBM PC/XT! TRS*80 II! EPSON QX10! ZENITH Z-100!

If you've been "patient" with slow disk drives for too long, SemiDisk will relieve your suffering.

Fast-acting.

The SemiDisk, a super-fast disk emulator, stores and retrieves data much faster than either a floppy or hard disk.

Easy to apply.

Installation is as easy as plugging the SemiDisk into an empty slot of your computer, and running the installation software provided.

Regular and extra-strength.

SemiDisk I is the standard model for S-100, SemiDisk II offers extra speed and flexibility for custom

S-100 applications.

Contains gentle buffers.

CP/M®80 installation software includes SemiSpool, which buffers print data in the SemiDisk. This allows the computer to be ready for other uses immediately after issuing a print command.

No emulator amnesia.

The optional Battery Backup Unit (BBU) plugs into the SemiDisk, and supplies power even when the computer is off. A battery keeps the data alive during power outages of four hours or more.

Stops head-aches.

Unlike a hard disk, which can 'crash' its head on the rotating disk

surface, and a floppy, which grinds the disk constantly, the SemiDisk gives you ultra-fast, silent data transfer.

And SemiDisk's price won't raise your blood pressure.

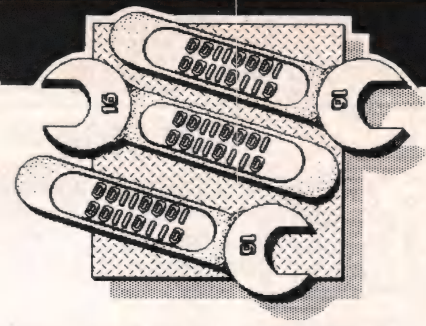
	512K	1Mbyte
SemiDisk I, S-100	\$995	\$1795
SemiDisk II, S-100	\$1245	\$2095
SemiDisk, TRS-80 II	\$995	\$1795
SemiDisk, IBM PC	\$945	\$1795
SemiDisk, Epson QX10	\$995	
SemiDisk, BBU	\$150	

SEMIDISK

SemiDisk Systems, Inc.
P.O. Box GG,
Beaverton, Oregon 97075
503-642-3100

Call 503-646-5510 for CBBS/NW and 503-775-4838 for CBBS/PCS, both SemiDisk-equipped computer bulletin boards (300/1200 baud). SemiDisk, SemiSpool trademarks of SemiDisk Systems. CP/M trademark of Digital Research.

Circle no. 62 on reader service card.



by Ray Duncan

The IBM PC/AT

IBM has confounded its critics and finally lowered the boom on "PC-compatible" manufacturers by releasing its next generation of high-end personal computers. From the hardware point of view, the PC/AT turned out to be pretty much as expected. It is conservatively designed with a 80286 CPU running at only 6 MHz, a 20-megabyte hard disk, 80-track 1-megabyte floppy disk drives, and 512K of RAM standard. As far as price goes, the PC/AT is clearly a "fighting machine"—only \$6000 out the door, just about the cost of a fully equipped and much slower PC/XT only a few months ago.

Performance-wise, the PC/AT has a very snappy feel. A preliminary computation-bound benchmark (generating the first 1000 primes) that I ran in PC/Forth yielded a time of 24.7 seconds on the "old" PC and 8.9 seconds on the PC/AT. This seems to verify the claims in the trade rags of a threefold speed advantage over the older machine. Real speed freaks will want to plug in an 80287 numeric coprocessor for number crunching; this chip is still pretty scarce and costs about \$375 (compared to \$175 for the 8087).

Physically, the PC/AT is slightly larger and heavier than the PC/XT. The keyboard has been extensively reorganized. The "Enter" key is now enormous, the "Esc" and "Backslash" keys have been moved into the upper right corner, and the "Caps Lock" key has the little LED indicator that everyone has been crying for. Most important of all, a keyboard lock now allows executives to go out to lunch, leave their PC/AT turned on, and not worry about anyone stealing their Lotus 1-2-3 worksheets.

The degree of upward hardware compatibility on the PC/AT from the previous PC is nothing short of astonishing. The PC/AT has a dual-nature

bus that will work either with the old expansion cards or with the new family of expansion options that exploit the 16-bit wide data path. The programmer's view of the hardware-software interface is, for all practical purposes, identical to the old PC. My own company's PC/Forth product, which drives the 6845 video controller and video refresh buffer directly and is extremely hardware dependent, runs on the PC/AT without a hitch.

The PC/AT's Technical Reference Manual is already available and contains plenty of required and/or interesting reading. Systems-level programmers particularly will want to inspect the sections on the dual DMA controllers, the RT/CMOS RAM, and the keyboard controller. The 150-page (!) BIOS contains some instructive examples of 80286 programming, which previously have been hard to come by. The Intel book *iAPX 286 Programmer's Reference Manual* is also helpful here.

The PC/AT is delivered with PCDOS 3.0, which appears to be a sort of warmed-over and expanded version of PCDOS 2.0. Support for partitioning the hard disk has been improved considerably. Also, the FORMAT command has been idiot-proofed, making it less easy to accidentally format the hard disk and erase all your files. PCDOS 3.0 adds a few new DOS commands (ATTRIB and LABEL), a few new CONFIG file commands, some support for file sharing in a networking environment... nothing very dramatic. However, IBM clearly has signaled its intention to move the PC/AT into the international market with standard DOS commands to redefine the keyboard and time/date display formats for the major European languages.

DEBUG and LINK don't appear to have changed at all. A RAM-DISK driver is now provided as a standard

utility. Also provided, sadly enough, is the same old horrible EDLIN that we first met in PCDOS 1.0.

To ensure upward compatibility with software written for the previous models of IBM PC, PCDOS 3.0 runs the 80286 in "Real Address Mode" rather than in the more advanced "Protected Virtual Address Mode." In Real Mode, the 80286 essentially functions like a slightly enhanced 8086 and has the same 1-megabyte address space. We can only hope that the alternative operating systems soon to appear for the PC/AT, such as Xenix, will exploit the sophisticated support available in Protected Mode for multitasking, virtual memory, and memory protection—thereby wringing all the available computing power out of the 80286.

Periscope PC Debugger

Brett Salter was kind enough to send me an advance copy of his "Periscope" hardware/software debugger for the IBM PC. This is an inexpensive, easy to use, and deceptively simple-looking program analysis tool. Periscope consists of a "short" board containing 16K of protected RAM and some additional logic, a cable with a pushbutton, and a disk containing a debugger/control program. Using Periscope, you can push the button at any time to grab control away from an executing program, after which you may search memory, disassemble programs, set breakpoints, or display memory and registers.

A particularly nice feature of Periscope is the ability to set breakpoints on a specific iteration of a loop, an access to a memory location, the contents of a memory location, or the contents of a register. The program can read the standard symbol tables generated by the PCDOS Linker, allowing the user

to set breakpoints either by name or by location. Dual monitors are supported when present, facilitating the debugging of graphics routines.

The manual is particularly well done, with clear installation instructions, a tutorial, detailed descriptions of all debugger commands, notes on debugging theory and techniques, a summary of all error messages, and a quick reference card. I continue to be astonished at how far the general level of documentation quality has risen since I built my first Imsai 8080 micro-computer. Periscope is available for \$295 from Data Base Decisions, 14 Bonnie Lane, Atlanta, GA 30328 (404) 256-3860.

Grandson of Floating-Point Benchmark

I thought I was rid of the Savage Floating Point Benchmark subject forever after the publication of the expanded table of results in the August 1984 *DDJ*. However, that table only incited *DDJ*'s loyal readers to send in even more timings. I guess I'll keep collecting for yet another while and publish the table again in early 1985. We're still looking for a COBOL listing, guys! Latest impressive mainframe result: 0.016 seconds with error 5E-7, Unix Basic Compiler, double precision, VM/UTS operating system, Amdahl 5860 CPU.

Counting Cycles

Bob Smith of Qualitas Inc. wrote in to chide me for a foolish statement I made in another column. I noted that, according to the Intel documentation, the sequence

```
SHL    BX,1
SHL    BX,1
SHL    BX,1
SHL    BX,1
```

occupied 8 bytes and required 8 cycles to execute, while the sequence

```
MOV    CL,4
SHL    BX,CL
```

which looks more elegant on paper, occupied only 4 bytes but required 28 cycles. Mr. Smith comments:

"The reference to instruction ti-

mings... prompts me to write about this counter-intuitive topic. It's tempting to rely solely upon the published raw instruction timings; however, the picture is more complicated. Additional effects, such as draining and refilling the 8088's four-byte pre-fetch instruction queue, can play an overriding role.

"As it turns out, the two-instruction combination `MOV CL,4...SHL BX,CL` is actually faster than four `SHL BX,1` instructions, although the raw instruction timings suggest the opposite. The reason appears to be that, for the four `SHL BX,1` instructions, their small size (2 bytes) and short execution time (2 clocks) drain the queue quickly—so quickly, in fact, that their execution time plus the time spent refilling the queue exceeds the time for the two-instruction version.

"These timings can be measured directly with a high-resolution software timer as described in our article 'Life in the Fast Lane,' *PC Tech Journal*, Vol. 1, No. 7 (April 1984). Please note that in the article the code segments in the listings of Figures 4 and 5 were inadvertently switched.

MSDOS Programming Pearl of the Month

When writing applications for MSDOS 2.0, many programmers have been disappointed in the display speeds available through the standard MSDOS output calls and have resorted to direct control of the hardware for better performance. It turns out that you can obtain substantially increased display speeds for "well-behaved" application programs on many MSDOS 2.0 systems simply by setting the "raw output mode" bit in the driver's device information word. This bit tells MSDOS not to check for a CTRL-C from the keyboard between each character it transfers to the output device. Listing One (page 100) provides the necessary code to set the raw output mode bit. Thanks to Dave Angel of Wang for this tip.

DOS 2.0 Filters

As a followup to August's "Clean" filter, this month we are printing the "TK" tokenizing filter contributed by Jim Mott of Sunnyvale, California (Listing Two, page 100). A detailed

LISP FOR THE IBM PERSONAL COMPUTER.

THE PREMIER LANGUAGE
OF ARTIFICIAL
INTELLIGENCE FOR
YOUR IBM PC.

DATA TYPES

- Lists and Symbols
- Unlimited Precision Integers
- Floating Point Numbers
- Character Strings
- Multidimensional Arrays
- Files
- Machine Language Code

MEMORY MANAGEMENT

- Full Memory Space Supported
- Dynamic Allocation
- Compacting Garbage Collector

FUNCTION TYPES

- EXPR/FEXPR/MACRO
- Machine Language Primitives
- Over 190 Primitive Functions

IO SUPPORT

- Multiple Display Windows
- Cursor Control
- All Function Keys Supported
- Read and Splice Macros
- Disk Files

POWERFUL ERROR RECOVERY

8087 SUPPORT

COLOR GRAPHICS

LISP LIBRARY

- Structured Programming Macros
- Editor and Formatter
- Package Support
- Debugging Functions
- .OBJ File Loader

RUNS UNDER PC-DOS 1.1 or 2.0

IQLISP

5 1/4" Diskette
and Manual _____ \$175.00
Manual Only _____ \$ 30.00

Integral Quality

P.O. Box 31970
Seattle, Washington 98103-0070
(206) 527-2918

Washington State residents add sales tax.
VISA and MASTERCARD accepted.
Shipping included for prepaid orders.

description of the operation of the filter with examples is contained in the listing itself. Jim writes:

"One warning: If you use the TK filter after the FIND filter with too little space on the %PIPE## disk, it will fail without any error messages. FIND doesn't seem to check the number of bytes written compared to the number requested. If the disk is full, FIND just stops writing to its pipe. It doesn't even tell you that something is wrong. When TK starts, the input file is only half there, so it doesn't have all the data to work with. If anyone would like to get a copy of the TK program on disk, tell them to send along \$15 to cover disk, postage, and handling." DDJ

Reader Ballot
Vote for your favorite feature/article.
Circle Reader Service No. 197.

16-Bit (Text begins on page 98)

Listing One

Selecting raw output mode for standard output under MSDOS 2.0 disables DOS's check for CTRL-C pending on the keyboard between each character transmitted to the standard output, thereby speeding up displays.

```
;
; Select Raw Output Mode on Standard Output Handle
;
mov     bx,1           ; i/o control read for
mov     ax,4400h       ; "device information"
int     21h
mov     dh,0           ; set upper byte of DH = 0
or      dl,20h         ; set raw mode bit in DL
mov     bx,1           ; i/o control write of
mov     ax,4401h       ; "device information"
int     21h
```

End Listing One

16-Bit

Listing Two

```
1          PAGE    55,132
2          TITLE   TK - Token Parsing Filter
3
4          ;
5          ; TK --- A Simple Token Parsing Filter for DOS 2.0
6          ;
7          ; (C) Copyright 1984 by Jim Mott
8          ;           3710 Slopeview Drive
9          ;           Sunnyvale, CA. 95148
10         ;           (408) 274-2620
11         ; All rights reserved. Permission granted to use this software for
12         ; personal, noncommercial purposes only.
13         ;
14         ;
15         ; This program is designed to be a filter for DOS 2.0.
16         ; It will tokenize its input and allow subsetting and/or
17         ; single token per line output.
18         ;
19         ; The format of the command is:
20         ;
21         ; TK {/RJx | /LJx} {/0} {/v} | {/v/v}
22         ; where /RJx means right justify all tokens to x positions
23         ;       /LJx means left justify all tokens to x positions
24         ;       In the above two entries x must be in [1..15]
25         ;       /0 means output one token per line
26         ;       /v means select token v for output. You may select any number,
27         ;       up to 255, of tokens to output. Repeats are allowed and you
28         ;       may change the order of the input tokens on the output line.
29         ;
30         ; For example, to extract a list of users from a VM directory file and write
31         ; a sorted list of them without passwords to the printer the following
32         ; command line would be used.
33         ;
34         ; FIND "USER " < DIRECT.VM | TK/LJ8/2/4/5/6/7/8/9 | SORT > PRN
35         ;
36         ;
37         ; For example, to find a list of all sub directories of the current directory
38         ; sorted by sub directory name we could use the following command line.
39         ;
```



```

40 ; DIR | FIND "<DIR>" | TK/LJ8/1/3/4 | SORT | MORE
41 ;
42 ;
43 ; For example, to generate a sorted list of all words used in a document with
44 ; one word per line we could use the following line.
45 ;
46 ; TK/RJ8/0 < FOOBAR.DOC | SORT | MORE
47 ;
48 ;
49 ; A note on comments and spelling. I am a very poor speller so I
50 ; take the view that programs can either well documented with
51 ; creatively spelled words or not documented with perfectly spelled words.
52 ; I choose the creative approach and am satisfied when the opcodes are
53 ; spelled correctly.
54 ;
55 ;
56 0000 STACK SEGMENT PARA STACK 'STACK'
57 0000 08 [ DB 8 DUP('Jim Mott (408) 274-2620')
58 4A 69 6D 20
59 4D 6F 74 74
60 20 20 28 34
61 30 38 29 20
62 32 37 34 2D
63 32 36 32 30
64 ]
65
66 00C0 STACK ENDS
67 ;
68 ;
69 0000 DSECT SEGMENT PARA 'DATA'
70 0000 FF [ BUFFER DB 255 DUP('?') ;WHERE TO PUT THE DATA
71 3F
72 ]
73
74 00FF 20 DB ' ' ;BE SURE TO END SCAN CORRECTLY
75 ;
76 0100 0000 GLEN DW 0 ;LENGTH OF GBUFF
77 0102 0104 R GBPTR DW GBUFF ;POINT TO START OF BUFFER
78 0104 FF [ GBUFF DB 255 DUP('G') ;BUFFER USED BY BUFGET
79 47
80 ]
81
82 ;
83 0203 ?? FLAG1 DB ? ;OPTIONS ENABLED
84 = 0001 F1RJ EQU 01H ;RIGHT JUSTIFY TOKENS
85 = 0002 FILJ EQU 02H ;LEFT JUSTIFY TOKENS
86 = 0004 F1ONE EQU 04H ;OUTPUT ONE TOKEN PER LINE
87 = 0008 F1SUB EQU 08H ;SUBSTRING FUNCTION REQUESTED
88 = 0010 F1WORK EQU 10H ;FILL TRAILING SPACES
89 = 0020 F1OERR EQU 20H ;ERROR IN OPTIONS STRING
90 = 0040 F1EOF EQU 40H ;END OF FILE ON STANDARD INPUT DEVICE
91 = 0080 F1QEOF EQU 80H ;QUE THE END OF FILE
92 ;
93 0204 ?? SPACES DB ? ;NUMBER OF TRAILING SPACES REQUIRED
94 ;
95 0205 ?? TOKSIZ DB ? ;TOKEN SIZE IF (F1RJ OR FILJ)
96 ;
97 0206 03 THREE DB 3 ;LENGTH OF EACH ENTRY
98 0207 ?? TOKCNT DB ? ;COUNT OF TOKENS IN TABLE
99 0208 02FD [ TOKTBL DB 3*255 DUP('0') ;TABLE OF TOKEN POINTERS AND LENGTHS
100 30
101 ]
102
103 ;
104 0505 ?? OUTCNT DB ? ;NUMBER OF SUBSETTING ENTRIES IN OUTERS
105 0506 FF [ OUTERS DB 255 DUP('1') ;LIST OF TOKEN NUMBERS TO OUTPUT
106 31
107 ]
108
109 ;
110 0605 ??? TOKPTR DW ? ;POINTER TO FREE TOKEN SPACE
111 0607 0384 [ TOKENS DB 900 DUP('2') ;STRING SPACE OF TOKENS
112 32
113 ]

```

(Continued on next page)

Listing Two

```

114
115
116      098B 54 4B 3A 20 49 6E      ; MSGVER DB 'TK: Incorrect DOS version. Must be at least 2.00.'
117      63 6F 72 72 65 63
118      74 20 44 4F 53 20
119      76 65 72 73 69 6F
120      6E 2E 20 20 4D 75
121      73 74 20 62 65 20
122      61 74 20 6C 65 61
123      73 74 20 32 2E 30
124      30 2E
125      098D 0D 0A 24                DB 0DH,0AH,'$'
126      09C0 54 4B 3A 20 49 6E      OPTMSG DB 'TK: Incorrect parameters given.'
127      63 6F 72 72 65 63
128      74 20 70 61 72 61
129      6D 65 74 65 72 73
130      20 67 69 76 65 6E
131      2E
132      09DF 0D 0A                DB 0DH,0AH
133      = 0021                    OPTMGL EQU $-OPTMSG
134      09E1 54 4B 3A 20 4E 6F      NOROOM DB 'TK: No room for user on device.'
135      20 72 6F 6F 6D 20
136      66 6F 72 20 75 73
137      65 72 20 6F 6E 20
138      64 65 76 69 63 65
139      2E
140      0A00 0D 0A                DB 0DH,0AH
141      = 0021                    LNOROOM EQU $-NOROOM
142      0A02 20                    CHRSPA DB ' ' ;A SPACE TO OUTPUT
143      0A03 0D 0A                CHRCLF DB 0DH,0AH ;<cr><lf> SEQUENCE
144      0A05                      DSECT ENDS
145
146      ;
147      ;
148      CSECT SEGMENT PARA 'CODE'
149      ASSUME CS:CSECT,DS:DSECT,SS:STACK
150      MAIN PROC FAR
151      0000 1E                    PUSH DS ;SET UP A RETURN ADDRESS
152      0001 2B C0                SUB AX,AX ;WE WANT TO RETURN TO DS:0000
153      0003 50                    PUSH AX
154      0004 B8 ---- R            MOV AX,DSECT ;POINT TO START OF DATA AREA
155      0007 8E D8                MOV DS,AX ;MAKE ASSUME AND REALITY AGREE
156      0009 B4 30                MOV AH,30H ;GET DOS VERSION NUMBER
157      000B CD 21                INT 21H ;CALL OS TO GET IT
158      000D 3C 02                CMP AL,2 ;IS IT AT LEAST 2.00
159      000F 7D 09                JNL MAIN00 ;YES - GOOD ENOUGH
160      0011 8D 16 098B R          LEA DX,MSGVER ; NO - POINT TO THE BAD DOS VERSION MESSAGE
161      0015 B4 09                MOV AH,9 ; DOS 1.?? FUNCTION TO PRINT AN ERROR
162      0017 CD 21                INT 21H ;CALL THE OPERATING SYSTEM
163      0019 C8                    RET ;AND DO A LONG RETURN
164      001A E8 006A R            MAIN00: CALL OPTIONS ;PARSE THE OPTIONS (AT ES:80) AND SET FLAGS
165      001D 8C D8                MOV AX,DS ;MAKE ES AND DS THE SAME NOW
166      001F 8E C0                MOV ES,AX ;SO THE STRING MOVES WORK NICELY
167      0021 F6 06 0203 R 20        TEST FLAG1,F10ERR ;WAS THERE AN ERROR IN THE OPTIONS?
168      0026 74 10                JZ MAIN01 ; NO - THEN GO WITH THIS BABY
169      0028 8D 16 09C0 R          LEA DX,OPTMSG ;YES - POINT TO OPTIONS ERROR MESSAGE
170      002C B9 0021                MOV CX,OPTMGL ; GET LENGTH OF MESSAGE
171      002F B8 0002                MOV BX,2 ; ERROR OUTPUT DEVICE HANDLE
172      0032 B4 40                MOV AH,40H ; OPERATING SYSTEM FUNCTION NUMBER
173      0034 CD 21                INT 21H ; CALL THE OPERATING SYSTEM
174      0036 EB 29                JMP SHORT MAIN03 ; AND RETURN AS DONE
175      0038 C6 06 0207 R 00        MAIN01: MOV TOKCNT,0 ;WE DON'T HAVE ANY TOKENS IN THE TABLE
176      003D 8D 06 0607 R          LEA AX,TOKENS ;POINT TO START OF TOKEN WORK AREA
177      0041 A3 0605 R            MOV TOKPTR,AX ;SAVE POINTER TO NEXT FREE BYTE IN TOKEN SPACE
178      0044 E8 027F R            CALL BUFGET ;READ IN A BUFFER
179      0047 F6 06 0203 R 40        TEST FLAG1,F1EOF ;IS THERE ANY DATA IN THE READ BUFFER
180      004C 75 13                JNZ MAIN03 ; NO - WE ARE DONE WITH THIS PROGRAM THEN
181      004E 49                    DEC CX ;YES - IGNORE THE TRAILING <cr>

```



```

181 004F 7E E7 JLE MAIN01 ;IF LENGTH IS ZERO OR WORSE JUST GET NEXT LINE
182 0051 8D 1E 0000 R LEA BX,BUFFER ;POINT TO THE FIRST BYTE OF THE DATA
183 0055 E8 0103 R MAIN02: CALL NEXTOK ;GET THE NEXT TOKEN
184 0058 0B C9 OR CX,CX ;ARE WE DONE WITH THIS LINE YET?
185 005A 75 F9 JNZ MAIN02 ; NO - GET YET ANOTHER TOKEN
186 005C E8 01A2 R CALL WRITE ;YES - WRITE THE LINE(S)
187 005F EB D7 JMP SHORT MAIN01 ;AND LOOP FOR THE NEXT LINE
188 0061 E8 0213 R MAIN03: CALL CRLF ;WRITE A FINAL <cr><lf> SEQUENCE
189 0064 B0 00 MOV AL,0 ;THIS IS THE RETURN CODE TO POST
190 0066 B4 4C MOV AH,4CH ;TERMINATE A PROCESS CODE
191 0068 CD 21 INT 21H ;END THIS PROGRAM
192 006A MAIN ENDP
193 ;
194 ;
195 ; OPTIONS - This subroutine will parse the options passed to the program and
196 ; set the required bits in FLAG1. No registers are preserved since
197 ; we are called only once, before the program has really started.
198 006A OPTIONS PROC NEAR
199 006A C6 06 0505 R 00 MOV OUTCNT,0 ;INITIALIZE OUTERS COUNT
200 006F BE 0081 MOV SI,81H ;POINT TO THE FIRST CHARACTER IN PARMS
201 0072 26: 8A 04 OPT01: MOV AL,BYTE PTR ES:[SI] ;GET A BYTE FROM PARM STRING
202 0075 46 INC SI ;POINT TO NEXT BYTE
203 0076 3C 0D CMP AL,0DH ;IS IT THE END OF THE STRING?
204 0078 75 01 JNE OPT02 ; NO - GOODY, MORE DATA TO PROCESS
205 007A C3 RET ;YES - RETURN TO CALLER THEN
206 007B 3C 20 OPT02: CMP AL,' ' ;ALLOW SPACES ANYWHERE BEFORE SLASHES
207 007D 74 F3 JE OPT01 ;IGNORE THEN THOUGH
208 007F 3C 2F CMP AL,'/' ;WE HAVE TO START WITH A SLASH NOW
209 0081 74 06 JE OPT04 ;IF IT IS A SLASH THEN PROCESS IT
210 0083 80 0E 0203 R 20 OPTERR: OR FLAG1,F10ERR ;OTHERWISE SET THE OPTIONS ERROR FLAG
211 0088 C3 RET ;AND RETURN
212 0089 26: 8A 04 OPT04: MOV AL,BYTE PTR ES:[SI] ;GET THE NEXT CHARACTER AFTER SLASH
213 008C 46 INC SI

```

(Continued on next page)

CDE SOFTWARE

CHECKS & BALANCES \$74.95

At last! A full-screen editing checking and finance program so informative and easy to use you'll wonder how you lived without it! Checks & Balances provides a single entry system, unlimited categories, flexible search and printing capability, optional check printing, and easy-to-understand commands in plain English. There is even an on-screen *HELP* function. Available for CP/M-80 2.2, MS-DOS and PC-DOS. (Price without check printing, \$54.95.)

ORIGINAL ADVENTURE \$19.95

CDE offers the fastest running version of this classic computer game available for a micro-computer.

CATCHUM and LADDER \$19.95

Now you can enjoy the excitement of arcade games on your CP/M computer with these two variations of popular games.

For a limited time, all three games—Adventure, Catchum and Ladder—are available for one low price of only **\$29.95!** Catch the fun!

Visa and MasterCard accepted. Write or phone for our complete catalogue of games, business and development software. Our telephone—(213) 661-2031.

2463 McCready Avenue • Los Angeles, CA 90039

Circle no. 8 on reader service card.

QUALITY SOFTWARE AT REASONABLE PRICES

CP/M Software by
Poor Person Software

Poor Person's Spooler \$49.95

All the function of a hardware print buffer at a fraction of the cost. Keyboard control. Spools and prints simultaneously.

Poor Person's Spread Sheet \$29.95

Flexible screen formats and BASIC-like language. Pre-programmed applications include Real Estate Evaluation.

Poor Person's Spelling Checker \$29.95

Simple and fast! 33,000 word dictionary. Checks any CP/M text file.

aMAZEing Game \$29.95

Arcade action for CP/M! Evade goblins and collect treasure.

Crossword Game \$39.95

Teach spelling and build vocabulary. Fun and challenging.

Mailing Label Printer \$29.95

Select and print labels in many formats.

Window System \$29.95

Application control of independent virtual screens.

All products require 56k CP/M 2.2 and are available on 8" IBM and 5" Northstar formats, other 5" formats add \$5 handling charge. California residents include sales tax.

Poor Person Software

3721 Starr King Circle
Palo Alto, CA 94306
tel 415-493-3735

CP/M is a registered trademark of Digital Research

Circle no. 51 on reader service card.

Listing Two

```

214 008D 3C 61          CMP     AL,'a'          ;IS IT LOWER CASE OR FUNNY?
215 008F 7C 02          JL      OPT4A         ; NO - PROCESS NORMALLY THEN
216 0091 2C 20          SUB     AL,'a'-'A'       ;YES - MAP LOWER CASE TO UPPER
217 0093 3C 4C          OPT4A: CMP    AL,'L'         ;MIGHT IT BE LEFT JUSTIFY OR NUMERIC
218 0095 7C 42          JL      OPTNUM        ;PERHAPS NUMERIC - CHECK IT OUT
219 0097 75 0E          JNE     OPT05         ;IT IS NOT LJ FOR SURE
220 0099 80 0E 0203 R 02 OR      FLAG1,F1LJ       ;ASSUME IT IS LJ FOR THE MOMENT
221 009E F6 06 0203 R 01 TEST    FLAG1,F1RJ       ;MAKE SURE THIS ISN'T A DUPLICATE
222 00A3 75 0E          JNZ     OPTERR        ;IF RJ ALREADY THEN BIG PROBLEMS
223 00A5 EB 10          JMP     SHORT OPT06      ;AND REJOIN COMMON JUSTIFY CODE
224 00A7 3C 52          OPT05: CMP    AL,'R'         ;MIGHT IT BE RIGHT JUSTIFY (RJ)?
225 00A9 75 08          JNE     OPTERR        ; NO - THEN IT IS AN ERROR
226 00AB 80 0E 0203 R 01 OR      FLAG1,F1RJ       ;YES - ASSUME FOR THE MOMENT IT IS
227 00B0 F6 06 0203 R 02 TEST    FLAG1,F1LJ       ;MAKE SURE WE AREN'T TRYING TO LEFT JUSTIFY TOO
228 00B5 75 CC          JNZ     OPTERR        ;IF WE ARE, THEN WE ARE IN DEEP S..T
229 00B7 26: 8A 04      OPT06: MOV    AL,BYTE PTR ES:0[S1] ;GET THE NEXT CHARACTER
230 00BA 46             INC     SI
231 00BB 3C 4A          CMP     AL,'J'          ;IS IT THE J WE EXPECT?
232 00BD 74 04          JE      OPT6A         ;YES - PROCESS IT NORMALLY THEN
233 00BF 3C 6A          CMP     AL,'j'          ;IS IT A LOWER CASE J?
234 00C1 75 C0          JNE     OPTERR        ; NO - THAT'S TO BAD.
235 00C3 26: 8A 04      OPT6A: MOV    AL,BYTE PTR ES:0[S1] ;GET THE FIRST BYTE OF THE NUMBER
236 00C6 46             INC     SI
237 00C7 E8 0247 R      CALL    DECBIN         ;IS IT A NUMBER?
238 00CA 72 B7          JC      OPTERR        ; NO - THEN WE HAVE AN ERROR
239 00CC 0A C0          OR      AL,AL          ;IS THE FIELD SIZE 0?
240 00CE 74 B3          JE      OPTERR        ;YES - IT IS IN ERROR THEN
241 00D0 3C 0F          CMP     AL,15           ;IS FIELD SIZE MORE THAN 15?
242 00D2 7F AF          JG      OPTERR        ;YES - IT IS IN ERROR THEN
243 00D4 A2 0205 R      MOV     TOKSIZ,AL       ;SAVE THE FIELD SIZE FOR JUSTIFIED FIELDS
244 00D7 EB 99          JMP     SHORT OPT01      ;AND PROCESS FURTHER OPTIONS
245 00D9 E8 0247 R      OPTNUM: CALL   DECBIN         ;IS IT A NUMBER AFTER SLASH?
246 00DC 72 A5          JC      OPTERR        ; NO - THEN IT IS AN ERROR
247 00DE 0A C0          OR      AL,AL          ;ZERO IS SPECIAL
248 00E0 75 07          JNE     OPT08         ;NOT ZERO - SAVE IT IN ARRAY THEN
249 00E2 80 0E 0203 R 04 OR      FLAG1,F1ONE       ;ZERO MEANS ONE TOKEN PER LINE
250 00E7 EB 89          JMP     SHORT OPT01      ;PROCESS SOME OTHER TOKEN THEN
251 00E9 2B C9          OPT08: SUB     CX,CX         ;GET A ZEROED DOUBLE REGISTER
252 00EB 8A 0E 0505 R      MOV     CL,OUTCNT       ;GET OFFSET INTO OUTERS FOR THIS GUY
253 00EF 8D 1E 0506 R      LEA     BX,OUTERS       ;POINT JUST BEFORE LIST OF OUTERS
254 00F3 03 D9          ADD     BX,CX           ;BX POINTS TO ORIGIN 1 SAVE SPOT
255 00F5 8B 07          MOV     BYTE PTR [BX],AL ;SAVE THE TOKEN POSITION TO WRITE
256 00F7 FE 06 0505 R      INC     OUTCNT         ;ADD ONE TO OUT COUNT
257 00FB 80 0E 0203 R 08 OR      FLAG1,F1SUB       ;MAKE SURE SUBSTITUTE FLAG IS ON
258 0100 E9 0072 R      JMP     OPT01          ;AND PLAY IT AGAIN SAM
259 0103          OPTIONS ENDP
260          ;
261          ;
262          ; NEXTOK - This subroutine will find the next token in the string pointed to
263          ; by BX, with length contained in CX, and move it to the end of the
264          ; token space. An entry in TOKTBL will be created for this token.
265          ; When the subroutine returns CX will be zero if the source data
266          ; string is empty. BX will point to the first character past the last
267          ; token.
268 0103          NEXTOK PROC NEAR
269 0103 8B 3E 0605 R      MOV     DI,TOKPTR       ;GET POINTER TO WHERE TO PUT TOKEN
270 0107 8A 07          NEXT01: MOV    AL,BYTE PTR 0[BX] ;LOOP PAST JUNK
271 0109 3C 20          CMP     AL,' '         ;IS IT A LEADING SPACE?
272 010B 75 04          JNE     NEXT03         ; NO - THEN WE HAVE A TOKEN
273 010D 43             INC     BX           ;YES - POINT TO THE NEXT CHARACTER
274 010E E2 F7          LOOP    NEXT01        ; AND TRY THAT ONE
275 0110 C3             RET                     ;RETURN IF WE ARE DONE WITH THE OUTPUT
276 0111 8B F3          NEXT03: MOV    SI,BX     ;SAVE POINTER TO START OF TOKEN
277 0113 84 01          MOV     AH,1           ;INITIAL GUESS FOR TOKEN LENGTH IS 1
278 0115 43             NEXT04: INC     BX           ;POINT TO NEXT CHARACTER IN INPUT STRING
279 0116 8A 07          MOV     AL,BYTE PTR [BX] ;GET THE CHARACTER
280 0118 3C 20          CMP     AL,' '         ;IS IT THE END OF THE TOKEN?
281 011A 74 06          JE      NEXT05         ;YES - WE HAVE SOME GOOD NUMBERS TO WORK WITH

```


282	011C	FE C4	INC	AH	: NO - INCREMENT COUNT OF CONTIGUOUS CHARACTERS
283	011E	E2 F5	LOOP	NEXT04	:CONTINUE TILL OUT OF CHARS OR A FIELD SEPARATOR
284	0120	FE CC	DEC	AH	:WE SHOULDN'T GET HERE BUT CORRECT FOR IT ANYWAY
285	0122	51	NEXT05: PUSH	CX	:SAVE NUMBER OF CHARACTERS LEFT IN SOURCE STRING
286	0123	F6 06 0203 R 03	TEST	FLAG1,F1RJ+F1LJ	:DO WE HAVE A MAXIMUM TOKEN LENGTH?
287	0128	74 2C	JZ	NEXT09	: NO - JUST A NORMAL TOKEN WRITE THEN
288	012A	3A 26 0205 R	CMP	AH,TOKSIZ	:YES - IS THIS TOKEN JUST RIGHT?
289	012E	74 26	JE	NEXT09	:IT SURE IS, WE WILL KEEP IT AS IS
290	0130	7C 06	JL	NEXT06	:IF TOKEN SIZE < MAX TOKEN SIZE WE MUST PAD
291	0132	8A 26 0205 R	MOV	AH,TOKSIZ	:OTHERWISE TAKE MAX TOKEN SIZE AS OUR OWN
292	0136	EB 1E	JMP	SHORT NEXT09	:AND CONTINUE NORMALLY
293	0138	A0 0205 R	NEXT06: MOV	AL,TOKSIZ	:GET THE TOKEN SIZE WE MUST PAD TO
294	013B	2A C4	SUB	AL,AH	:AL NOW CONTAINS NUMBER OF SPACES NEEDED
295	013D	F6 06 0203 R 02	TEST	FLAG1,F1LJ	:LEFT JUSTIFY THE THING? (PAD RIGHT WITH SPACE)
296	0142	74 0A	JZ	NEXT07	: NO - MUST PAD TO THE LEFT WITH SPACES
297	0144	A2 0204 R	MOV	SPACES,AL	:YES - SAVE HOW MANY SPACES TO FILL WITH
298	0147	80 0E 0203 R 10	OR	FLAG1,F1WORK	: MARK AS WORK TO DO LATER ON
299	014C	EB 08	JMP	SHORT NEXT09	: AND JOIN MAINLINE CODE
300	014E	8A C8	NEXT07: MOV	CL,AL	:CX CONTAINS THE NUMBER OF LEADING SPACES
301	0150	C6 05 20	NEXT08: MOV	BYTE PTR [DI],	: PUT A LEADING SPACE IN THIS TOKEN
302	0153	47	INC	DI	:POINT TO THE NEXT SLOT
303	0154	E2 FA	LOOP	NEXT08	:AND FILL IN ALL NEEDED SPACES
304	0156	8A CC	NEXT09: MOV	CL,AH	:CX NOW CONTAINS TOTAL NUMBER OF CHARS IN TOKEN
305	0158	FC	CLD		:MAKE THE DIRECTION EVER UPWARD
306	0159	F3/ A4	REP	MOVSB	:MOVE THE TOKEN TO ITS SPOT
307	015B	F6 06 0203 R 10	TEST	FLAG1,F1WORK	:IS IT LEFT JUSTIFIED (WE NEED SPACES)
308	0160	74 0F	JZ	NEXT11	: NO - WE ARE DONE WITH THE HARD PART THEN
309	0162	8A 0E 0204 R	MOV	CL,SPACES	:GET COUNT OF SPACES NEEDED
310	0166	C6 05 20	NEXT10: MOV	BYTE PTR [DI],	: MOVE IN A TRAILING SPACE
311	0169	47	INC	DI	:POINT TO NEXT SLOT
312	016A	E2 FA	LOOP	NEXT10	:AND CONTINUE TILL ALL TRAILING SPACES DONE
313	016C	80 26 0203 R EF	AND	FLAG1,255-F1WORK	:RESET THE WORK TO DO BIT
314	0171	8B 16 0605 R	NEXT11: MOV	DX,TOKPTR	:GET POINTER TO START OF THIS TOKEN

(Continued on next page)

Thanks to YOU We're Growing
with YOU and your Computer . . .



LEO ELECTRONICS, INC.
P.O. Box 11307
Torrance, CA. 90510-1307
Tel: 213/212-6133 800/421-9565
TLX: 291 985 LEO UR

**We Offer . . . PRICE . . . QUALITY . . .
PERSONAL SERVICE**

64K UPGRADE

9 Bank (IBM PC)	\$43.65	(150ns)
	\$41.85	(200ns)
4164 (150ns)	\$4.85 ea.	
(200ns)	\$4.65 ea.	
8 Bank (other PC)	\$38.80	(150ns)
	\$37.20	(200ns)
4164 (150ns)	\$4.85 ea.	
(200ns)	\$4.65 ea.	

256K "Mother-Saver" Upgrade

256K - (150ns)	\$36.00 ea.	
6116P-3 -	\$4.40	2732 - \$3.95
2716 -	\$3.20	2764 - \$7.00
TMS-2716 -	\$4.95	27128 - \$24.00

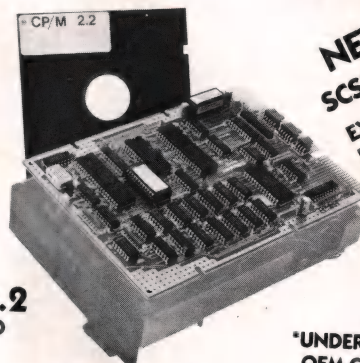
We accept checks, Visa, Mastercard or Purchase Orders
from qualified firms and institutions. U.S. Funds only.
Call for C.O.D. California residents add 6 1/2% tax.
Shipping is UPS. Add \$2.00 for ground and \$5.00 for
air. All major manufacturers. All parts 100% guaranteed.
Pricing subject to change without notice.

Circle no. 44 on reader service card.

Dr. Dobb's Journal, November 1984

The Little Board™ ... \$349*

The world's simplest and least expensive CP/M computer



**NEW
SCSI/PLUS™
EXPANSION
I/O OPTION**

**CP/M 2.2
INCLUDED**

***UNDER \$200 IN
OEM QUANTITIES**

- 4 MHz Z80A CPU, 64K RAM, Z80A CTC, 2732 Boot ROM
- Mini/Micro Floppy controller (1-4 Drives, Single/Double Density, 1-2 sided, 40/80 track)
- Only 5.75 x 7.75 inches, mounts directly to a 5 1/4" floppy drive
- 2 RS232C Serial Ports (75-9600 baud & 75-38,400 baud), 1 Centronics Printer Port
- Power Requirement: -5VDC at .75A; +12VDC at .05A/On-board -12V converter
- CP/M 2.2 BDOS • ZCPR3 CCP • Enhanced AMPRO BIOS
- AMPRO Utilities included:
 - read/write to more than 2 dozen other formats (Kaypro, Teletype, IBM CP/M86....)
 - format disks for more than a dozen other computers
 - menu-based system customization
- BIOS and Utilities Source Code Available
- SCSI/PLUS Adapter:
 - Mounts directly to Little Board
 - Slave I/O board control
 - Full ANSC X379.2
 - 16 bidirectional I/O lines
 - \$99/Quantity 1

AMPRO
COMPUTERS, INCORPORATED

Distributor/Dealer/Reps
Inquiries Invited

67 East Evelyn Ave. • Mountain View, CA 94041 • (415) 962-0230 • TELEX 4940302

Circle no. 3 on reader service card.

Listing Two

```

315 0175 89 3E 0605 R      MOV     TOKPTR,DI      ;SAVE POINTER TO NEXT FREE TOKEN BYTE
316 0179 F6 06 0203 R 03   TEST     FLAG1,F1LJ+F1RJ ;DO WE HAVE FIXED LENGTH TOKENS?
317 017E 74 04           JZ      NEXT12      ; NO - TAKE THEM AS WE GET THEM
318 0180 8A 26 0205 R      MOV     AH,TOKSIZ      ;YES - SET THIS TOKENS LENGTH TO WHATEVER
319 0184 8A CC           NEXT12: MOV     CL,AH      ;SAVE LENGTH OF TOKEN
320 0186 80 03           MOV     AL,3      ;NUMBER OF BYTES PER ENTRY
321 0188 F6 26 0207 R      MUL     TOKCNT      ;AX IS NOW AN OFFSET IN TOKTBL
322 018C 8D 36 0208 R      LEA     SI,TOKTBL      ;POINT TO START OF TOKEN TABLE
323 0190 03 F0           ADD     SI,AX      ;SI POINTS TO AN ENTRY IN TOKEN TABLE
324 0192 88 0C           MOV     BYTE PTR [SI],CL ;MOVE IN LENGTH OF ENTRY
325 0194 89 54 01         MOV     WORD PTR 1[SI],DX ;SAVE POINTER TO START OF TOKEN IN TABLE
326 0197 FE 06 0207 R      INC     TOKCNT      ;COUNT ONE MORE TOKEN
327 019B 59             POP     CX      ;CX CONTAINS NUMBER OF SOURCE CHARACTERS LEFT
328 019C 0B C9           OR      CX,CX      ;DONE YET?
329 019E 74 01           JZ      NEXT13      ;YES - RETURN
330 01A0 49             DEC     CX      ; NO - CORRECT FOR UNDERCOUNTING BY ONE
331 01A1 C3           NEXT13: RET      ;AND RETURN
332 01A2           NEXTOK ENDP
333
334
335 ; WRITE - This routine will write the tokens to the standard output device.
336 ; It is controlled by the settings of flags in FLAG1.
337 01A2           WRITE PROC NEAR
338 01A2 2B C9           SUB     CX,CX      ;GET AN EMPTY LOOP COUNTER
339 01A4 8A 0E 0207 R      MOV     CL,TOKCNT      ;CL CONTAINS TOTAL NUMBER OF TOKENS READ
340 01A8 0B C9           OR      CX,CX      ;DO WE HAVE ANYTHING TO WRITE OUT?
341 01AA 75 01           JNZ     WRITE1      ;YES - THEN GO FOR IT
342 01AC C3           RET      ; NO - WE ARE DONE BEFORE WE EVEN BEGIN
343 01AD F6 06 0203 R 08   WRITE1: TEST     FLAG1,F1SUB ;ARE WE CHANGING THEIR ORDER?
344 01B2 75 08           JNZ     WRITE3      ;YES - THEN USE DIFFERENT WRITE LOGIC
345 01B4 2A D2           SUB     DL,DL      ; NO - JUST OUTPUT THEM ALL IN ORDER
346 01B6 E8 01E2 R      WRITE2: CALL     TOUT      ;WRITE THE SUCKER
347 01B9 FE C2           INC     DL      ;POINT TO THE NEXT TOKEN
348 01BB E2 F9           LOOP    WRITE2      ;AND GO THROUGH THEM ALL
349 01BD EB 18           JMP     SHORT WRITE6 ;RETURN, A JOB WELL DONE
350 01BF 8A 0E 0505 R      WRITE3: MOV     CL,OUTCNT ;GET THE NUMBER OF TOKENS TO WRITE
351 01C3 8D 1E 0506 R      LEA     BX,OUTERS      ;POINT TO FIRST ONE TO OUTPUT
352 01C7 8A 17           WRITE4: MOV     DL,BYTE PTR [BX] ;GET A TOKEN TO WRITE
353 01C9 3A 16 0207 R      CMP     DL,TOKCNT      ;IS IT LESS THAN OR EQUAL TO MAX TOKEN?
354 01CD 7F 05           JG      WRITE5      ; NO - DON'T WRITE IT THEN
355 01CF FE CA           DEC     DL      ;YES - ADJUST FOR ORIGIN ONE AND
356 01D1 E8 01E2 R      CALL     TOUT      ; WRITE THIS TOKEN THEN
357 01D4 43           WRITE5: INC     BX      ;POINT TO THE NEXT TOKEN COUNT TO WRITE
358 01D5 E2 F0           LOOP    WRITE4      ;AND LOOP THROUGH THE WHOLE LIST
359 01D7 F6 06 0203 R 04   WRITE6: TEST     FLAG1,F1ONE ;ARE WE OUTPUTTING ONE TOKEN PER LINE?
360 01DC 75 03           JNZ     WRITE7      ;YES - THE LAST <cr><lf> HAS BEEN WRITTEN
361 01DE E8 0213 R      CALL     CRLF      ; NO - WRITE A TRAILING <cr><lf> SEQUENCE
362 01E1 C3           WRITE7: RET      ;DONE, GO HOME NOW
363 01E2           WRITE ENDP
364
365 ;
366 ;
367 ; TOUT - This routine will find and write the token from the input line that is
368 ; in position DL on that line.
369 ;
370 ;
371 TOUT PROC NEAR
372 PUSH     BX      ;SAVE THE REGISTERS
373 PUSH     CX
374 PUSH     DX
375 MOV     AL,3      ;NUMBER OF BYTES PER TOKTBL ENTRY
376 MUL     DL      ;GET AN OFFSET INTO TOKTBL FOR THIS ONE
377 LEA     BX,TOKTBL ;POINT TO START OF TABLE
378 ADD     BX,AX      ;NOW POINT TO THE CORRECT 3 BYTE ENTRY
379 SUB     CX,CX      ;ZERO THE COUNTER
380 MOV     CL,BYTE PTR [BX] ;GET THE NUMBER OF CHARACTERS IN THIS TOKEN
381 MOV     DX,WORD PTR 1[BX] ;AND POINT TO FIRST BYTE OF SAID TOKEN
382 CALL     OSWRITE ;WRITE TO THE STANDARD OUTPUT DEVICE
383 CALL     FLAG1,F1ONE ;ONLY ONE TOKEN PER LINE?
384 TEST

```

(Continued on page 10)

AVAILABLE BACK ISSUES

1982	1983	1984
No. 64—Feb.	No. 75—Jan.	No. 87—Jan.
No. 66—April	No. 76—Feb.	No. 88—Feb.
No. 68—June	No. 77—March	No. 89—March
No. 69—July	No. 78—April	No. 90—April
No. 70—Aug.	No. 80—June	No. 91—May
No. 71—Sept.	No. 81—July	No. 92—June
No. 72—Oct.	No. 82—Aug.	No. 93—July
No. 73—Nov.	No. 83—Sept.	No. 94—Aug.
No. 74—Dec.	No. 84—Oct.	No. 95—Sept.
	No. 85—Nov.	No. 96—Oct.
	No. 86—Dec.	

TO ORDER: send \$3.50 per issue to: Dr. Dobb's Journal,
2464 Embarcadero Way, Palo Alto, CA 94303.

Name _____

Address _____

City _____

State _____

Zip _____

3024

Circle no. 84 on reader service card.

DeSmet C

**8086/8088
Development
Package \$109**

FULL DEVELOPMENT PACKAGE

- Full K&R C Compiler
- Assembler, Linker & Librarian
- Full-Screen Editor
- Execution Profiler
- Complete **STDIO** Library (>120 Func)

Automatic DOS 1.X/2.X SUPPORT

BOTH 8087 AND SOFTWARE FLOATING POINT

OUTSTANDING PERFORMANCE

- First and Second in AUG '83 BYTE benchmarks

SYMBOLIC DEBUGGER \$50

- Examine & change variables by name using C expressions
- Flip between debug and display screen
- Display C source during execution
- Set multiple breakpoints by function or line number

DOS LINK SUPPORT \$35

- Uses DOS .OBJ Format
- LINKs with DOS ASM
- Uses Lattice® naming conventions

Check: ☐ Dev. Pkg (109)
☐ Debugger (50)
☐ DOS Link Supt. (35)

SHIP TO: _____

ZIP _____

CWARE
CORPORATION

P.O. BOX C
Sunnyvale, CA 94087
(408) 720-9696

All orders shipped UPS surface on IBM format disks.
Shipping included in price. California residents add
sales tax. Canada shipping add \$5, elsewhere add
\$15. Checks must be on US Bank and in US Dollars.
Call 9 a.m. - 1 p.m. to CHARGE by VISA/MC/AMEX.

Circle no. 18 on reader service card.

**FAT
directory
sector spacing
copy protection
sector id's
formats**

- See what's really on your disks.
- Change formats.
- Correct bad tracks.
- Restore deleted files.
- Runs on floppies and hard disks.

Disk Explorer

Price \$75 U.S. funds
Check or credit card.
No charge for mailing.

Quaid Software Limited

45 Charles Street East, Sixth Floor
Toronto, Ontario, Canada M4Y 1S2

Telephone: (416) 961-8243

Circle no. 89 on reader service card.

Listing Two

```

383 01FE 74 05          JZ      TOUT02      : NO - WRITE A SPACE THEN
384 0200 E8 0213 R      CALL     CRLF        : YES - WRITE A <cr><lf> SEQUENCE
385 0203 E8 0A          JMP      SHORT TOUT03 : AND RETURN
386 0205 B9 0001      TOUT02: MOV     CX,1      : LENGTH OF SPACE IS ONE
387 0208 8D 16 0A02 R   LEA      DX,CHRSPA    : POINT TO A SPACE
388 020C E8 0226 R      CALL     OSWRITE     : WRITE TO STANDARD OUTPUT DEVICE
389 020F 5A          TOUT03: POP      DX        : RESTORE THE REGISTERS
390 0210 59          POP      CX
391 0211 5B          POP      BX
392 0212 C3          RET              : AND RETURN
393 0213          TOUT      ENDP
394
395
396
397 ; CRLF - Everybody knows what this routine does.
398
399 0213          CRLF      PROC      NEAR
400 0213 50          PUSH     AX          : SAVE THE REGISTERS
401 0214 53          PUSH     BX
402 0215 51          PUSH     CX
403 0216 52          PUSH     DX
404 0217 B9 0002      MOV      CX,2      : LENGTH OF CR LF STRING
405 021A 8D 16 0A03 R   LEA      DX,CHRCRLF    : POINT TO THE DATA TO WRITE
406 021E E8 0226 R      CALL     OSWRITE     : WRITE TO STANDARD OUTPUT DEVICE
407 0221 5A          POP      DX        : RESTORE THE REGISTERS
408 0222 59          POP      CX
409 0223 5B          POP      BX
410 0224 58          POP      AX
411 0225 C3          RET              : AND RETURN
412 0226          CRLF      ENDP
413
414
415 ; OSWRITE - This routine will write the characters pointed to by DS:DX, of
416 ;           length contained in CX, to the standard output file. If any errors
417 ;           are detected a message will be written to the standard error device
418 ;           and flag F1E0J will be set.
419
420 0226          OSWRITE   PROC      NEAR
421 0226 BB 0001      MOV      BX,1      : HANDLE OF STANDARD OUTPUT DEVICE
422 0229 B4 40      MOV      AH,40H     : WRITE TO FILE OR DEVICE SYSTEM FUNCTION
423 022B CD 21      INT      21H       : CALL THE OPERATING SYSTEM
424 022D 72 04      JC      OSWR01     : IF ERROR 5, OR 6 THEN END
425 022F 3B C8      CMP      CX,AX     : DID WE WRITE AS MANY CHARACTERS AS WE WANTED?
426 0231 74 13      JE      OSWR99     : RETURN IF ALL WENT WELL
427 0233 8D 16 09E1 R   OSWR01: LEA      DX,NOROOM    : POINT TO THE NO SPACE MESSAGE
428 0237 B9 0021      MOV      CX,LNOROOM : GET THE LENGTH OF THE MESSAGE
429 023A B8 0002      MOV      BX,2      : GET HANDLE FOR STANDARD ERROR DEVICE
430 023D B4 40      MOV      AH,40H     : WRITE TO FILE OR DEVICE SYSTEM FUNCTION
431 023F CD 21      INT      21H       : LET HIM KNOW WE ERRED
432 0241 80 0E 0203 R 40 OR      FLAG1,F1E0F    : PRETEND EOF ON INPUT DEVICE SO PROGRAM STOPS
433 0246 C3          OSWR99: RET              : AND RETURN
434 0247          OSWRITE   ENDP
435
436 ; DECBIN - On entry this routine has the first character to convert to binary
437 ;           in AL. SI points to additional characters. On exit SI points to
438 ;           the first non-numeric character found. AL contains the binary value
439 ;           and Carry isn't set. If Carry is set on return then an invalid
440 ;           number was found.
441 0247          DECBIN   PROC      NEAR
442 0247 53          PUSH     BX          : SAVE A REGISTER
443 0248 E8 0273 R      CALL     DECB04     : CHECK FOR NUMERIC IN AL
444 024B 73 03      JNC      DECB02     : IF AL WAS NUMERIC IT IS NOW IN [0..9]
445 024D F9          DECB01: STC              : MAKE SURE CARRY FLAG IS SET
446 024E 5B          POP      BX        : RESTORE THE REGISTERS
447 024F C3          RET              : AND RETURN INDICATING AN ERROR
448 0250 8A D8      DECB02: MOV      BL,AL      : GET THE TOTAL SO FAR
449 0252 26: 8A 04      MOV      AL,BYTE PTR ES:0[SI] : GET A BYTE FROM THE INPUT STREAM
450 0255 46          INC      SI

```



```

451 0256 E8 0273 R      CALL DEC804      ;CHECK IT FOR NUMERIC
452 0259 73 06          JNC DEC803      ;IF IT IS NUMERIC THEN JUGGLE SOME
453 0258 8A C3          MOV AL,BL      ;OTHERWISE GET THE VALUE TO RETURN
454 025D F8            CLC            ;CLEAR THE CARRY BIT TO SAY IT WORKED
455 025E 4E            DEC SI         ;MAKE SURE NEXT CHAR TO GET IS NON-NUMERIC
456 025F 5B            POP BX        ;RESTORE THE REGISTER
457 0260 C3            RET           ;AND RETURN
458 0261 8A F8          DEC803: MOV BH,AL ;SAVE THE NUMBER FOR A MINUTE
459 0263 80 0A          MOV AL,10     ;GET THE BASE
460 0265 F6 E3          MUL BL        ;SHIFT CURRENT COUNT LEFT ONE POSITION (BASE AL)
461 0267 8A DF          MOV BL,BH     ;MAKE BX A GOOD NUMBER
462 0269 2A FF          SUB BH,BH     ;BX NOW CONTAINS 16 BIT VALUE OF ASCII DIGIT
463 026B 03 C3          ADD AX,BX     ;ADD IN THE LATEST DIGIT
464 026D 0A E4          OR AH,AH      ;MAKE SURE NO OVERFLOW
465 026F 75 DC          JNE DEC801    ;IF THERE WAS THIS IS AN ERROR
466 0271 EB DD          JMP SHORT DEC802 ;CONTINUE ON OUR WAY
467 0273 2C 30          DEC804: SUB AL,'0' ;IS IT LESS THAN A NUMBER
468 0275 7C 06          JL DEC805     ;YES - RETURN WITH CARRY SET
469 0277 3C 09          CMP AL,9      ;IS IT MORE THAN A NUMBER
470 0279 7F 02          JG DEC805     ;YES - RETURN WITH CARRY SET
471 027B F8            CLC            ; NO - MAKE SURE CARRY IS OFF
472 027C C3            RET           ; THEN RETURN THE NUMBER
473 027D F9            DEC805: STC     ;SET CARRY ON
474 027E C3            RET           ;AND RETURN
475 027F              DECBIN ENDP
476
477
478 ; BUFGET - This routine will read one 'line' from the standard input device to
479 ; BUFFER. On exit CX contains the count of characters read. F1EOF is
480 ; set if an end of file condition is encountered.
481 027F              BUFGET PROC NEAR
482 027F 50              PUSH AX        ;SAVE THE REGISTERS

```

(Continued on next page)

**Hello,
I'm Bill Salkin**

As editor of **PC FIRING LINE/PC UNDERGROUND**, the bi-monthly technical disk magazine for the IBM PC, I invite you to join us as we discuss critical



error-handling routines, bicubic splines, drool over "printf" source code, share **FREEWARE**, provide programmer-oriented **DEMOS** and tips, dissect utilities like the DOS 2.x **EXEC** function, continue our **ADA**, **ASM**, **C**, **FORTH**, and **LISP** tutorials, and more! Sizzling ready-to-use source code included in each jam-packed issue (currently 2 DS/DD disks per issue!).

- Price: \$12 per issue, or \$72 for a one-year (six-issue) subscription. Make checks payable to **ABComputing**. (All currency in U.S. dollars. Foreign countries send money orders and add \$5 airmail for each issue.)
- Make non-profit copies for your friends. Those receiving these copies are asked to pay us \$6 to help defray our production costs. Make checks payable to **ABComputing**.
- Requires 128K RAM, any release of **DOS**, and one double-sided disk drive. Note: We ship **ONLY DS/DD** disks.

ABComputing

Dept. 50, P.O. Box 5503, North Hollywood, CA 91616-5503
(818) 509-9002

Circle no. 1 on reader service card.

Dr. Dobb's Journal Subscription Problems?

No Problem!



Give us a call and we'll straighten it out. Today.

Outside California
CALL TOLL FREE: 800-321-3333

Inside California
CALL: 619-485-6535 or 6536

Circle no. 85 on reader service card.

Listing Two

```

483 0280 53          PUSH    BX
484 0281 52          PUSH    DX
485 0282 57          PUSH    DI
486 0283 56          PUSH    SI
487 0284 F6 06 0203 R 80  TEST    FLAG1,F1EOF    ;SHOULD WE REFLECT AN IMMEDIATE EOF?
488 0289 74 0C          JZ      BUFG00    ; NO - STANDARD LOGIC HERE THEN
489 0288 80 0E 0203 R 40  OR      FLAG1,F1EOF    ;YES - SET THE END OF FILE BIT
490 0290 80 26 0203 R 7F  AND     FLAG1,255-F1EOF ;AND SAY IT IS NO LONGER PENDING
491 0295 E8 41          JMP     SHORT BUFRET    ;RETURN NOW
492 0297 2B C9          BUFG00: SUB    CX,CX    ;COUNT OF CHARACTERS GOTTEN
493 0299 8D 3E 0000 R    LEA     DI,BUFFER    ;POINT DESTINATION TO BUFFER
494 029D E8 02DE R      BUFG01: CALL   CGET     ;GET ONE CHARACTER
495 02A0 F6 06 0203 R 40  TEST    FLAG1,F1EOF    ;DID WE GET END OF FILE ON THAT TRY?
496 02A5 74 15          JZ      BUFG02    ;YES - LETS HOPE IT IS AN ERROR
497 02A7 0A C9          OR      CL,CL    ;IS THERE ANYTHING IN THE BUFFER?
498 02A9 74 2D          JZ      BUFRET    ; NO - JUST RETURN WITH CX=0 AND F1EOF
499 02AB B4 0D          MOV     AH,0DH    ;YES - SLAP A <cr> ON THE END
500 02AD E8 031C R      CALL    CPUT     ; PUT IT AT END OF BUFFER
501 02B0 80 26 0203 R 8F  AND     FLAG1,255-F1EOF ; CLEAR THE END OF FILE BIT
502 02B5 80 0E 0203 R 80  OR      FLAG1,F1EOF    ; SAY NEXT TIME TURN ON EOF FOR SURE
503 02BA E8 1C          JMP     SHORT BUFRET    ; AND RETURN THIS LAST BUFFER
504 02BC 80 FC 0D          BUFG02: CMP    AH,0DH    ;IS IT THE RECORD TERMINATOR CHARACTER?
505 02BF 74 07          JE      BUFG03    ;YES - DON'T TURN THAT INTO A SPACE
506 02C1 80 FC 20          CMP     AH,20H    ; NO - IF NOT <cr> AND < 20H THEN MAKE A SPACE
507 02C4 7D 02          JGE     BUFG03    ;IF >=20H THEN USER AS IT
508 02C6 B4 20          MOV     AH,' '    ;OTHERWISE MAKE IT A SPACE
509 02C8 E8 031C R      BUFG03: CALL   CPUT     ;WRITE THE CHARACTER TO THE OUTPUT BUFFER
510 02CB 80 FC 0D          CMP     AH,0DH    ;DID WE JUST WRITE THE RECORD TERMINATOR?
511 02CE 74 08          JE      BUFRET    ;YES - THEN RETURN
512 02D0 80 F9 FF          CMP     CL,255    ;HAVE WE WRITTEN 255 CHARACTERS YET?
513 02D3 75 C8          JNE     BUFG01    ; NO - GET THE NEXT CHARACTER THEN
514 02D5 C6 05 0D          MOV     BYTE PTR [DI],0DH ;MAKE IT A TERMINATOR
515 02D8 5E          BUFG01: POP     SI    ;RESTORE THE REGISTERS
516 02D9 5F          POP     DI
517 02DA 5A          POP     DX
518 02DB 58          POP     BX
519 02DC 58          POP     AX
520 02DD C3          RET      ;AND RETURN
521 02DE          BUFG02: ENDP
522          ;
523 02DE          CGET    PROC    NEAR
524 02DE 8B 16 0100 R      MOV     DX,GLEN    ;IS THERE ANY DATA IN GBUFF?
525 02E2 0B 02          OR      DX,DX    ;IF COUNT IS ZERO THEN THERE ISN'T
526 02E4 75 25          JNZ     CGET01    ;THERE IS DATA SO READ IT
527 02E6 51          PUSH    CX    ;SAVE THE REGISTERS WE MIGHT NEED
528 02E7 57          PUSH    DI
529 02E8 B4 3F          MOV     AH,3FH    ;OS FUNCTION TO READ FROM STANDARD DEVICE
530 02EA BB 0000          MOV     BX,0    ;FILE HANDLE TO READ FROM (STANDARD DEVICE)
531 02ED B9 00FF          MOV     CX,255    ;NUMBER OF CHARACTERS TO READ
532 02F0 8D 16 0104 R    LEA     DX,GBUFF    ;POINT TO WHERE TO PUT THE DATA
533 02F4 8B 16 0102 R    MOV     GBPTR,DX    ;SAVE POINTER TO FIRST CHARACTER
534 02F8 CD 21          INT     21H    ;CALL THE OPERATING SYSTEM TO HANDLE THE WORK
535 02FA A3 0100 R      MOV     GLEN,AX    ;SAVE THE NUMBER OF CHARACTERS READ
536 02FD 8B 00          MOV     DX,AX    ;PUT DATA COUNT IN DX WHERE IT BELONGS
537 02FF 5F          POP     DI    ;RESTORE THE REGISTERS
538 0300 59          POP     CX
539 0301 0B 02          OR      DX,DX    ;DID WE GET DATA OR EOF?
540 0303 75 06          JNZ     CGET01    ;DATA THIS TIME
541 0305 80 0E 0203 R 40  OR      FLAG1,F1EOF    ;SET THE END OF FILE ENCOUNTERED BIT
542 030A C3          RET      ;AND RETURN
543 030B 8B 1E 0102 R    CGET01: MOV    BX,GBPTR ;GET POINTER TO CHARACTER TO RETURN
544 030F 8A 27          MOV     AH,BYTE PTR [BX]
545 0311 FF 06 0102 R    INC     GBPTR    ;GET CHARACTER AND INCREMENT POINTER
546 0315 FF 0E 0100 R    DEC     GLEN    ;DECREMENT LENGTH
547 0319 FE C1          INC     CL    ;COUNT THIS CHARACTER
548 031B C3          RET      ;AND RETURN THE CHARACTER
549 031C          CGET    ENDP

```



```

550
551      031C          CPUT  PROC  NEAR
552      031C 88 25      MOV  BYTE PTR [DI],AH ;SAVE THE CHARACTER
553      031E 47        INC  DI              ;POINT TO NEXT SPOT
554      031F 80 F9 FF  CMP  CL,255         ;WILL WE OVERREACH NEXT TIME?
555      0322 75 01      JNE  CPUT01        ; NO - GOOD THING
556      0324 4F        DEC  DI              ;YES - CAN'T LET THAT HAPPEN
557      0325 C3        CPUT01: RET          ;ALL DONE
558      0326          CPUT  ENDP
559      0326          CSECT  ENDS
560
                    END  MAIN

```

Segments and groups:

Name	Size	align	combine	class
CSECT	0326	PARA	NONE	'CODE'
DSECT	0A05	PARA	NONE	'DATA'
STACK	00C0	PARA	STACK	'STACK'

Symbols:

Name	Type	Value	Attr
BUFFER	L BYTE	0000	DSECT Length =00FF
BUFG00	L NEAR	0297	CSECT
BUFG01	L NEAR	029D	CSECT
BUFG02	L NEAR	02BC	CSECT
BUFG03	L NEAR	02C8	CSECT
BUFGET	N PROC	027F	CSECT Length =005F
BUFRET	L NEAR	02D8	CSECT
CGET	N PROC	02DE	CSECT Length =003E
CGET01	L NEAR	0308	CSECT
CHRCLE	L BYTE	0A03	DSECT
CHRSPA	L BYTE	0A02	DSECT
CPUT	N PROC	031C	CSECT Length =000A
CPUT01	L NEAR	0325	CSECT
CRLF	N PROC	0213	CSECT Length =0013
DECB01	L NEAR	024D	CSECT
DECB02	L NEAR	0250	CSECT
DECB03	L NEAR	0261	CSECT
DECB04	L NEAR	0273	CSECT
DECB05	L NEAR	027D	CSECT
DECBIN	N PROC	0247	CSECT Length =0038
F1EOF	Number	0040	
F1LJ	Number	0002	
F1OERR	Number	0020	
F1ONE	Number	0004	
F1QEOF	Number	0080	
F1RJ	Number	0001	
F1SUB	Number	0008	
F1WORK	Number	0010	
FLAG1	L BYTE	0203	DSECT
GBPTR	L WORD	0102	DSECT
GBUFF	L BYTE	0104	DSECT Length =00FF
GLEN	L WORD	0100	DSECT
LNOROOM	Number	0021	
MAIN	F PROC	0000	CSECT Length =006A
MAIN00	L NEAR	001A	CSECT
MAIN01	L NEAR	0038	CSECT
MAIN02	L NEAR	0055	CSECT
MAIN03	L NEAR	0061	CSECT

(Continued on next page)

Listing Two

MSGVER	L BYTE 098B	DSECT	
NEXT01	L NEAR 0107	CSECT	
NEXT03	L NEAR 0111	CSECT	
NEXT04	L NEAR 0115	CSECT	
NEXT05	L NEAR 0122	CSECT	
NEXT06	L NEAR 0138	CSECT	
NEXT07	L NEAR 014E	CSECT	
NEXT08	L NEAR 0150	CSECT	
NEXT09	L NEAR 0156	CSECT	
NEXT10	L NEAR 0166	CSECT	
NEXT11	L NEAR 0171	CSECT	
NEXT12	L NEAR 0184	CSECT	
NEXT13	L NEAR 01A1	CSECT	
NEXTOK	N PROC 0103	CSECT	Length =009F
NOROOM	L BYTE 09E1	DSECT	
OPT01	L NEAR 0072	CSECT	
OPT02	L NEAR 0078	CSECT	
OPT04	L NEAR 0089	CSECT	
OPT05	L NEAR 00A7	CSECT	
OPT06	L NEAR 0087	CSECT	
OPT08	L NEAR 00E9	CSECT	
OPT4A	L NEAR 0093	CSECT	
OPT6A	L NEAR 00C3	CSECT	
OPTERR	L NEAR 0083	CSECT	
OPTIONS	N PROC 006A	CSECT	Length =0099
OPTMGL	Number 0021		
OPTMSG	L BYTE 09C0	DSECT	
OPTNUM	L NEAR 00D9	CSECT	
OSWR01	L NEAR 0233	CSECT	
OSWR99	L NEAR 0246	CSECT	
OSWRITE	N PROC 0226	CSECT	Length =0021
OUTCNT	L BYTE 0505	DSECT	
OUTERS	L BYTE 0506	DSECT	Length =00FF
SPACES	L BYTE 0204	DSECT	
THREE	L BYTE 0206	DSECT	
TOKCNT	L BYTE 0207	DSECT	
TOKENS	L BYTE 0607	DSECT	Length =0384
TOKPTR	L WORD 0605	DSECT	
TOKSIZ	L BYTE 0205	DSECT	
TOKTBL	L BYTE 0208	DSECT	Length =02FD
TOUT	N PROC 01E2	CSECT	Length =0031
TOUT02	L NEAR 0205	CSECT	
TOUT03	L NEAR 020F	CSECT	
WRITE	N PROC 01A2	CSECT	Length =0040
WRITE1	L NEAR 01AD	CSECT	
WRITE2	L NEAR 0186	CSECT	
WRITE3	L NEAR 01BF	CSECT	
WRITE4	L NEAR 01C7	CSECT	
WRITE5	L NEAR 01D4	CSECT	
WRITE6	L NEAR 01D7	CSECT	
WRITE7	L NEAR 01E1	CSECT	

Warning Severe
Errors Errors
0 0

End Listing

LYNX

the professional's replacement
for Microsoft L80

lynx



LYNX™ DOES

EVERYTHING L80™ DOES AND MORE!

LYNX™ LETS YOU USE ALL AVAILABLE MEMORY.

You can create COM files that are 10K larger—without overlays—by replacing L80 with LYNX.

LYNX™ IS A FULL OVERLAY LINKER

Running twice as fast as its nearest competitor, LYNX is tree structured, multi-segmented and multi-leveled, with automatic or explicit overlay invocation. You can run programs that are larger than available memory.

LYNX™ HAS BEEN HELPING MICROSOFT FORTRAN PROGRAMMERS FOR YEARS—NOW IT IS ALSO AVAILABLE FOR MICROSOFT BASIC AND AZTEC C.™

LYNX™ IS A QUALITY PRODUCT from the same company who offers you:

- GrafTalk, the business graphics package for Micros
- GRAPHICS development tools
- 2780/3780, 3270, X.25 communications
- MICRO TO MICRO communications



609 Main Street
Ridgefield, CT 06877
1-800-HELP RGI (203) 431-4661
Telex. 643351

\$250

LYNX and GrafTalk are trademarks of Redding Group, Inc. L80 is a trademark of Microsoft. AZTEC C is a trademark of MANX Software Systems.

Circle no. 60 on reader service card.

Starlight FORTH

for

65SC802 & 65SC816

The new 16 bit CMOS 6502's from
Western Design Center and GTE

- Apple
- Atari
- AIM 65
- Commodore
- Ohio Scientific

- Highly Extended fig-FORTH
 - Full Double Number Set - Strings
 - Optional Tools: Meta-Cross-Compiler with Assembler and Debugger
 - Multi-Tasking Super Fast FORTH-83 soon
- Your present 8 bit system will run faster
with 16 bit software (FORTH) and a 65SC802
and still runs your existing 8 bit software

Starlight FORTH Systems

15247 N. 35th St.

Phoenix, AZ 85032
(602) 992-5181

See us at the
FIG FORTH National Convention
November 16 & 17

Circle no. 69 on reader service card.

Changing Your Address?

Staple your label here.

To change your address, attach your address label from the cover of the magazine to this coupon and indicate your new address below.

Name _____

Address _____

Address _____ Apt. # _____

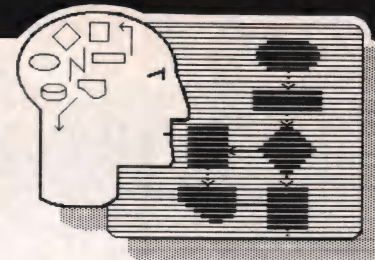
City _____ State _____

Zip _____

Mail to: DDJ, 2464 Embarcadero Way,
Palo Alto, CA 94303

A Realizable Fantasy

by Michael Swaine



In 1976, Jim Warren, the guiding spirit of this magazine's early days, defined in Jim Warren style the term *realizable fantasies*: "... projects that we feel (1) are within the bounds of current technology and knowledge, (2) can be implemented by members of the hobbyist community, and (3) can ... be realized within 24 months or less."

Warren asserted *DDJ*'s commitment to pursue and promote such realizable fantasies, and he pointed to their probable source: "You are part of this ... the proposals and designs and certainly the implementations will come from you. Send us your ideas, your creations, your problems, and your solutions, so that we may share them with everyone."

DDJ was a hobbyist newsletter in 1976, and today it's a magazine read by professional programmers, an evolution that closely parallels the evolution of the microcomputer software industry. One regrettable consequence of all this seriousness is a dearth of fantasy: little fantasy in *DDJ*, little fantasy in the industry, little fantasy in the design of new software. If you are keeping an eye on the bottom line, you aren't looking up.

Feeling as I did that Warren's spirit of playful creativity was ominously missing in software design in the mid-80s, I hit upon the simple plan of calling some active software designers and others who had thoughts about software design and asking them what tool of the trade they felt was missing: the realization of what fantasy would make the task of software design easier, more enjoyable, more accessible to a new generation of programmers, more manageable to those currently doing design work?

As I say, it was a simple plan—simple to the point of naivete as it turned out. I soon found, as I began polling people for their realizable fantasies,

that the question was not as open-ended as I thought. It appears to have one answer.

Here's Steve Dompier, software designer: "I want to program by flowcharting. I want to draw a flowchart and employ a tool that turns the flowchart into a program. It might produce source code or object code—it doesn't matter. We already have menus of icons; why not icons for flowchart symbols? Function boxes, decision boxes ... we could have icons for the various loop structures.

"Maybe the tool would work hierarchically: you'd start with a simple design then get more and more detailed. That implies the top-down design approach, which is the way at least some programmers work. And you could examine the structure you were building hierarchically: point to a box in the structure, zoom in on it, and it expands to show what's in it.

"Building at the lowest level, you might have to type some code, but you'd certainly have some library functions available, represented as icons, selectable by pointing. Looking at the program, you should be able to zoom down to a specific area, to a specified level, even right down to source code."

Here's Lee Felsenstein, hardware designer: "What I'd like to see—quite literally see—is the ability for the visual representation of structure, very much like a schematic diagram, but for software. I'd like to see software schematic diagrams realized in computer graphics. It's been done for the hardware with CAD systems."

It soon became clear that the ability to design by manipulating visual representations of program structures is something that very many designers want. That's the fantasy. Is it realizable? Sure. Should it be realized? Are you kidding? That this has not yet happened can perhaps be relegated to the

conventional-wisdom realm of explanation, subdomain barefoot progeny of cobblers.

The software design tool described here could be in the process of development now; someone reading this column could at this moment be working on it. Or could start work on it tomorrow or today. This minute.

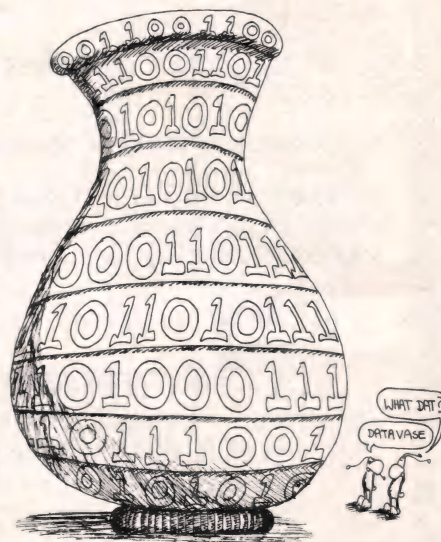
The challenge is implicit. But let's be blatant. Christmas is coming. The software industry, after all, has been good to a lot of people. It has given the world (or that part of it with access to computers at least) some very useful tools. Won't someone give the software industry a present?

(There's a second challenge implicit in all this. Despite my pretense that a single answer exists to the question of what software design tools are needed, fantasy is inherently unbounded. There are, of course, infinitely many realizable fantasies of software design. You are welcome, as ever, to send us yours.)

DDJ

Reader Ballot

Vote for your favorite feature/article.
Circle Reader Service No. 198.



©1984



**Subscribe to
Dr. Dobb's Journal
For The Holidays
And Give Twelve
Gifts in One!**

For Christmas, treat a friend (or yourself!) to a full year (12 issues) of Dr. Dobb's Journal, delivered monthly to the doorstep, and save \$5 off the newsstand price.

Better yet, subscribe for 2 years and save \$13!

Every month, you'll find articles on Small-C, FORTH, CP/M, S-100, Compiler optimization, Concurrent Programming and more.

Yes! Please start a subscription for
_____ 2 yrs. \$47 _____ 1 yr. \$25

for:

Name _____

Address _____

_____ Zip _____

_____ Charge my Visa,

MasterCard,

American Express

_____ I enclose a check/money
order

_____ Please bill me later

Name _____

Address _____

_____ Zip _____

Credit Card _____ Exp. date _____

Account No. _____

Signature _____

Dr. Dobb's Journal

P.O. Box 27809

San Diego, CA 92128

3025

Circle no. 86 on reader service card.

The Preferred C Compiler

"...C86 was the only compiler we tested that ran every benchmark we tried and gave the expected results... Computer Innovations

C86 was the compiler that our staff programmers used both before and six months after we conducted the tests."

J. Houston, BYTE MAGAZINE - February 1984

***FAST EXECUTION** -
of your programs.

***FULL & STANDARD
IMPLEMENTATION OF C** -
includes all the features described by
K & R. It works with the standard
MSDOS Linker and Assembler; many
programs written under UNIX can
often be compiled with no changes.

***8087 IN-LINE** -
highly optimized code provides 8087
performance about as fast as possible.

***POWERFUL OPTIONS** -
include DOS2 and DOS1 support and
interfaces; graphics interface capability;
object code; and librarian.

***FULL LIBRARY WITH SOURCE** -
6 source libraries with full source code
the "large" and "small" models, soft-
ware and 8087 floating point, DOS2
and DOSALL.

***FULL RANGE OF SUPPORT
PRODUCTS FROM COMPUTER
INNOVATIONS** -
including Halo Graphics, Phact File
Management, Panel Screen Manage-
ment, C Helper Utilities and our
newest C to dBase development
tool.

***HIGH RELIABILITY** -
time proven through thousands of
users.

***DIRECT TECHNICAL
SUPPORT** -
from 9 a.m. to 6 p.m.

Join The Professional Programmers Who Agree C86™ Is The C Compiler Of Choice

For Further Information Or To Order Call:

800-922-0169

Technical Support: (201) 542-5920

980 Shrewsbury Avenue
Suite PW509
Tinton Falls, NJ 07724



Computer Innovations, Inc.

C86™

PRICES SUBJECT TO CHANGE WITHOUT PRIOR NOTICE
UNIX IS A TRADEMARK OF BELL LABS C86 IS A TRADEMARK OF COMPUTER INNOVATIONS, INC. MSDOS IS A TRADEMARK OF MICROSOFT
PCDOS IS A TRADEMARK OF INTERNATIONAL BUSINESS MACHINES

Circle no. 13 on reader service card.

COMPUTER CALISTHENICS

by Michael Wiesenberg

Mid-morning at I-Q Industries in the heart of Silicon Valley. Grey Scrivener, senior technical writer specializing in software manuals, starts a fresh pot of coffee brewing at one of I-Q's permanent coffee stations. He glances up at the convex mirror mounted near the ceiling where the corridor intersects another. Seeing someone heading his way, Scrivener hurriedly adjusts the knot in his bottle-green knit tie.

Spotswood Gilbert, senior programmer, rounds the corner carrying a chocolate-covered doughnut. "Mornin', Grey. Sure glad you're makin' some *good* coffee. Can't stand that company stuff."

"Good morning, Spotswood. The coffee will be ready in a few minutes. How are you today?"

"Fine. How's the novel comin'?"

"I'm on the third rewrite."

Bob Levin, hotshot programmer, ratty sneakers squeaking, approaches the coffee machine. Scrivener casts a faintly disapproving glance at the young man's uncombed blond hair, trailing over a wrinkled Star Wars t-shirt. Levin stops. "How ya doing, Spotty? Hey, Grey."

Gilbert smiles. "Howdy, Bobby. You're up kinda early, aren't you?"

"I left early last night, so I thought I'd make it in for the morning doughnuts."

The water has dripped through. Scrivener fills Gilbert's cup and then his own. "By 'early' he means he left before midnight."

Sally McRae, another hotshot programmer, arrives wearing her usual "uniform": jeans, Nikes, and a Bay-to-Breakers t-shirt. "Hi, guys. You're here early, Bob. Don't tell me *you* made the coffee."

Levin looks embarrassed. "Nah, I don't know how to run that thing. Grey made it."

Scrivener sniffs. "Something's on fire."

McRae jumps. "What? Where?"

Marian Smith, system operator, appears, cigarette hanging out of her mouth. Scrivener fans the air with his hand to dissipate the smoke. Smith takes no more notice of this than she does the other criticism she frequently gets about her smoking. "Good morning, gentlemen, Sally. Who's on Goldfinger? I have to bring it down this afternoon. I'll be backing up all the system disks, but you'll have to take care of your own privates."

Levin eyes McRae appreciatively. "Playing two-man volleyball at noon, Sally?"

"Yes, and I don't see why it has to be called *two-man*. What's wrong with *two-person* volleyball?"

Scrivener always has the full story on word usage. "In a phrase like 'two-man volleyball' or a word like 'chairman,' the *man* has no gender. It refers to both men *and* women. When historians refer to the history of *man*, they are not excluding women, you know."

McRae looks him right in the eye. "Crap."

Gilbert looks uncomfortable. Where he's from, ladies don't talk that way. "Say, I got an interesting message on IQMAIL today. It was from my friend at HP, Jim Davis. You know, he's the one into puzzles. He gave me a great one. He said it was an old one he'd found that could be worked out by hand, but he thought a computer could do it a lot more efficiently. I wonder if you could write a program to solve it, Bob?"

"Let's hear it. I'll look at it after work."

McRae turned her scorn from Scrivener to Gilbert. "What makes you think *I* can't do it?"

"Well, sure, let's all give it a try. Here it is:

"There are two world-famed mathematicians, Mr. P and Mr. S. A friend

of theirs who likes puzzles tells both of them that he has two numbers in mind. The only thing he will tell both of them about these two numbers is that they are both integers, greater than 1, and they are not the same.

"He then whispers to Mr. P the product of the two numbers.

"He whispers to Mr. S the sum of the two numbers.

"Mr. P knows that Mr. S knows the sum, but he doesn't know what that sum is. Similarly, Mr. S knows that Mr. P knows the product, but he doesn't know what that product is.

"The following conversation then takes place:

"Mr. P: 'I don't know the numbers.'

"Mr. S: 'I know that; I also don't know the numbers.'

"Mr. P: 'Oh, then I know the numbers.'

"Mr. S: 'Really, then I do also.'

"Your task, of course, is to discover the two numbers. It would take you a long time to figure them out by hand, so write a program to do it. A *short* program."

Smith is more interested in IQMAIL than puzzles. She sets her cigarette down next to the coffee maker, its lit edge extending beyond the wood veneer top of the table, and pours herself another cup. "Is IQMAIL working for everybody? I've heard complaints from some of the other divisions."

Levin also pours a second cup. "I patched in my own communications package for a backup when IQMAIL goes down, which was twice yesterday. That's a terrific puzzle, Spots. The smallest two numbers that fit are 2 and 3, right?"

"Yes. You can see that their product is 6 and their sum is 5. But these two gentlemen are expert mathematicians. They would know that the numbers could *not* be 2 and 3."

"Mm hm. If the guy has told S that

the sum was 5, S would know right away what the two numbers were. And the sum couldn't be 6, either, come to think of it, because the two possibilities would be either 3 and 3, which are eliminated because they're the same, or 2 and 4, which would mean again that he knew the numbers. Tricky."

Scrivener starts another pot brewing. He begins sniffing again.

Smith notices that the glowing end of her cigarette is almost burning the table. She picks it up and jams it in her mouth.

Scrivener again waves his hands around. "I have a puzzle that I don't think can be solved by computer."

Levin slides the pot away before the water has stopped dripping, holding his cup in place to catch the diminishing stream. When his cup is filled, he adroitly slides it out of the way, and slips the pot back in place to catch the last drops. "Any puzzle can be solved by computer, Grey."

"Oh. Write me a program, then, that answers this question: What is a horse?"

McRae giggles. "We used to make up silly questions like that at Cal, with sillier answers. Why is a duck? Because one of its feet are both the same. When is a horse? When it has a leg in each corner."

All smile, Scrivener included. "Can you describe a horse programmatically? Maybe a few FOR-NEXT loops or, even better, a little recursion?"

Well, gentle readers, can you do it? Your programs must be short and elegant. They can be in any language, although I hope you don't pick something obscure like Snobol or SPL. The algorithms may be demonstrated in a good pseudolanguage if you wish or perhaps flowcharts. The best solution to each puzzle wins a DDJ t-shirt and will be published here. Remember that a good program is self-documenting.

DDJ

Reader Ballot

Vote for your favorite feature/article.
Circle Reader Service No. 199.

\$3.00 C Compiler

Due to popular demand Dr. Dobb's Journal has reprinted Ron Cain's C compiler from our sold-out 1980 issues #45 and #48. The reprint includes "A Small C Compiler for the 8080s" and "Runtime Library for the Small C Compiler" for only \$3.00, postage included. To Order:

Enclose \$3.00 for each copy with this coupon and send to:
Dr. Dobb's Journal, 2464 Embarcadero Way,
Palo Alto, CA 94303.

Please send _____ copy(ies) to:

name _____

address _____

city _____

state _____

zip _____

All reprint orders must be prepaid.

Circle no. 87 on reader service card.



**TOTAL
CONTROL
WITH LMI FORTH**

PC FORTH™
IBM PC & XT,
HP-150,
Macintosh,
Apple II,
CompuPro,
Sage & CP/M-68K,
Wang PC,
All CPM and
MSDOS computers.

Try the professional language offering the utmost performance in the shortest development time. Transport your applications between any of our enhanced 83-Standard compilers or expanded 32-bit versions. Choose from our wide selection of programming tools including native code compilers, cross-compilers, math coprocessor support, and B-Tree file managers. All fully supported with hotline, updates, and newsletters.

Laboratory Microsystems Incorporated
Post Office Box 10430, Marina del Rey, CA 90295
Phone credit card orders to (213) 306-7412



Circle no. 40 on reader service card.

New Release 1.8 – SOLID GOLD

CodeSmith™-86

B:\fabcode.COM CodeSmith-86

>

Also runs on some IBM-PC Compatibles

AX=8086	SP=8087	SS=1983
BX=0000	BP=0000	DS=1984
CX=0000	SI=0000	ES=1985
DX=1138	DI=0000	CS=2001
		IP=0001

PL	ZR	NC	NV	UP	NA	PE	EI
----	----	----	----	----	----	----	----

```

2001:0000  53          24  IO_INIT:  PUSH    BX                ;TAG A LINE
2001:0001  9BDEC2          FADDP   ST(2),ST
2001:0004  BB3100          MOV     BX,Offset VECTOR_TABLE_2
2001:0007  803E5E~      34      CMP     DOS_VERSION_NUM,'2' ;BREAKPOINT SET
2001:000C  7305          JAE     TRASH_IT
2001:000E  BB0100          MOV     BX,Offset VECTOR_TABLE_1
2001:0011  EB02          JMP     Short LONG_LABELS_ARE_OK_AS_YOU_LIKE
2001:0013  F2AB  00777  TRASH_IT:  REPNZ   STOSW          ;STOP 777th TIME
2001:0015          LONG_LABELS_ARE_OK_AS_YOU_LIKE:
2001:0015  8DAD63~      LEA     BP,WIERD_CODE + 2[DI]
2001:0019  240C          AND     AL,00011100B      ;CHANGE RADIX
2001:001B  45          DB     69
  
```

MEMORY DUMP																
>>DOS_VERSION_NUM																
Absolute Address=03C9E Segment Offset=03C4:005E																
1984:0050	41	53	43	49	49	20	53	55-50	50	4F	52	54	20	32	20	ASCII SUPPORT 2
1984:0060	20	2D	2D	2D	43	6F	64	65-53	6D	69	74	68	2D	38	36	-- CodeSmith-86
1984:0070	20	4D	41	4B	45	53	2D	44-45	42	55	47	47	49	4E	47	MAKES DEBUGGING
1984:0080	20	41	2D	42	4C	41	53	54-21	2D	2D	2D	2D	2D	2D	2D	A BLAST!

WNDW 2 LVL 1

It's here—THE Multi-Window Interactive Debugger that's STATE-OF-THE-ART.

- Scroll Up/Down thru full-screen disassemblies & memory dumps
- **Load** and **Write** Commands much easier, more powerful than DEBUG's
- "Snapshot" a complete debugging state onto disk—resume later
- True passpoints and execution path counters
- SCREENSAVE mode saves and restores user's graphic display when breakpoint hit
- Disassemble selected ranges of memory code to disk—compatible with IBM Assembler
- Stop on data Read/Write or memory range access

Hot-Line technical support

The Professional's Choice—CodeSmith-86

Multiple copies purchased by:

Lotus Development Corp., MicroPro, VisiCorp, IBM.

Requires MS-DOS & 160K RAM.

OEM and dealer inquiries invited.

VISUAL AGE

642 N. Larchmont Blvd. • Los Angeles, CA 90004

Circle no. 75 on reader service card.

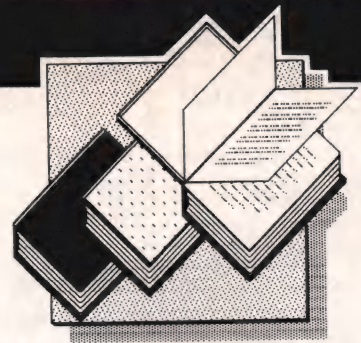
CodeSmith, TM International Arrangements, Inc.

MS, TM Microsoft Corp.

IBM, TM International Business Machines Corp.

CALL (213) 439-2414
\$14500

Send **VISUAL AGE - CodeSmith-86** 642 N. Larchmont Blvd. L.A., CA 90004 • (213) 439-2414
Name _____
Company _____
Address _____
City _____
Acct. No. _____
Exp. Date _____
State _____ Zip _____
☐ VISA ☐ MasterCard
☐ Payment Enclosed
☐ COD



Programming in C

by Stephen G. Kochan

Published by Hayden Book

Company

373 pages

Reviewed by Ian Ashdown

There are now as many books about the C programming language as you could possibly want—from simple tutorials about the basics of the language to a completely documented implementation of a C compiler . . . and of course, that definitive work, Kernighan & Ritchie's *The C Programming Language*.

C is a programming language noted for its sparseness and economy of expression. Most books on C are relatively slim, reflecting its powerful simplicity. Not so, however, *Programming in C*; at 373 pages, this book is nothing short of voluminous. Each subject is dealt with in exhaustive and ultimately exhausting detail. If you are in search of an answer about almost any aspect of C, you will find it in this book. You will also find it quickly—the index is thorough and well organized.

The promotional blurb on the cover reads: "A complete, easy-to-understand introduction to the C programming language for the novice and experienced programmer alike. Contains over 90 program examples and covers all the latest language features." Unlike most promotional claims, this is one that I fully agree with. The book starts with a very quick review of programming fundamentals, shows you how to write, compile, and execute a simple C program, and then is off and away into examining the language itself.

What do you want to know about? Variables, constants, data types, and arithmetic expressions? Program flow control? Arrays, functions, structures, character strings, pointers, bit operations, and bit fields? The preprocessor, perhaps, or separate compilation? In-

put and output or maybe miscellaneous and advanced topics such as unions, dynamic memory allocation, and the goto statement (more on this later)? *Programming in C* covers them all with many clear and detailed examples.

The exhaustive detail of this book can best be demonstrated by an example. In any language, how can one possibly find enough to say about the lowly if statement to fill 18 pages of text? Answer: Read this book and see.

I must remind myself, however, that I am reading the book as a fairly proficient C programmer. It wasn't too long ago that I taught myself the language with only K & R's book as a guide. It was a frustrating experience at times, compounded by that book's sparse literary style. For the neophyte, *Programming in C* is undoubtedly a far better book from which to learn the language.

Having learned C, I have found no better reference book for my tastes than K & R's *The C Programming Language*. There are days, however, when I simply cannot remember some of C's more cryptic features, and K & R does nothing to disperse the mental fog. Over the past month, I have found that *Programming in C*, despite the author's propensity for verbosity, serves very nicely. In other words, the book truly is "for the novice and experienced programmer alike."

A full C compiler running under Unix on a minicomputer is assumed throughout the book, with only passing references to the idiosyncracies of microcomputer versions. This approach has obvious advantages in that the book itself will stand as a valid reference manual for as long as the C language is in general use. The same cannot be said for some other books on C that are based on current versions of particular C compilers.

Other nice features of this book are its appendices. Appendix A is a summary of the language, written in a style that is both concise and clear. (This is in marked contrast to K & R's Appendix A, "The C Reference Manual." Their language summary is concise but rarely clear to the average reader.) Especially useful is a full range of examples involving pointers, with explanations in English.

Appendix B, entitled "Common Programming Mistakes," is unfortunately too short to be of much use. But, then, serious programming in C means that you likely will make every mistake possible—the power and conciseness of the language guarantees it. Learn by doing and have a good reference manual by your side at all times to rescue you.

Appendix E discusses a program that all microcomputer-bound C programmers dream of: "lint." The discussion, which is a short page and a half, merely covers what this program analyzer looks for when presented with C source code. It flags such potential problems as undeclared variables, unreachable code, improperly declared functions, invalid structure member accesses, possible operator precedence errors . . . sigh. What I wouldn't give to have the source code for this program!

K & R's book is often referred to as the standard reference manual for the language, and yet it does not cover some extensions that have since become standard for Unix-based C compilers. You can now pass structures as arguments (not just pointers to them) to functions, and a new data type called "void" explicitly tells the compiler that a particular function does not return any value. Also the "enumerated" data type, taking a cue from Pascal and other languages, allows a restricted range of values to be assigned to a variable. *Programming in C* is one of the few books that discusses

these features in detail. Now if only the microcomputer C compiler writers could take heed of these extensions . . .

The author does display some interesting programming prejudices. The goto, break, and continue statements are relegated to the back of the book in a chapter entitled "Miscellaneous Features and Advanced Topics." The reasoning is that these statements serve only to "obfuscate the program's logic and are not generally considered a part of good programming practice."

While I agree with the author that the goto statement is an anachronism that should be avoided whenever possible, I cannot agree with his argument about the break and continue statements. I find that these two statements serve to simplify and clarify the logic of program loops.

Also curious is the relegation of command line arguments to the same chapter. This to me is one of the most powerful features of C—it brings some of the grandeur of Unix to CP/M-based microcomputers, among other things. It also greatly extends the utility of programs that present the user with runtime options. It should have been given more space than its allotted page and a half.

Apart from that, I can say, while the book may be too wordy for my liking, it nevertheless can serve as an excellent reference on the C language as well as a tutorial for learning C. *Programming in C* may not be the ideal book for your needs, but by all means give it your consideration if you see it.

Pascal for Programmers

by Olivier Lecarme and

Jean-Louis Nebut

Published by McGraw-Hill

\$22.95, 272 pages

Reviewed by Dr. Joseph B.

Rothstein

The position that BASIC has long enjoyed as the language of choice among beginning programmers faces a serious challenge from Pascal. Designed by Kathleen Jensen and Niklaus Wirth in the early 1970s, the Pascal programming language was intended to implement the principles of structured programming that were emerging at that time.

Because the design of the language

encourages good programming habits, Pascal has been adopted for use in the first course in programming at many colleges and universities, and it is now available on most microcomputers as well.

Numerous texts use Pascal to introduce the principles of computers and programming to the neophyte, and some of them serve this audience well. However, there is a need for a text that would introduce and describe the entire Pascal language to someone already familiar with computer programming; it is this need that Lecarme and Nebut address in *Pascal for Programmers*. Although Pascal is well suited to beginners, it is also a powerful and elegant general-purpose language that has achieved considerable acceptance among business and scientific programmers as well.

Pascal for Programmers attempts to discuss each aspect of Pascal for experienced programmers—that is, those who don't need explanations of what computers, programs, algorithms, or expressions are. The authors have assumed that readers are already familiar with some high-level programming language, such as Fortran, BASIC, or PL/I, and that they understand the fundamental concepts of computer programming.

As a result, beginning programmers may find the text inappropriate and confusing, and the authors make it clear that the book isn't intended for use in an introductory course in computer programming. Rather it offers an exhaustive treatment of Pascal and a precise description of the language standardized by ISO (International Standards Organization). As such, it differs from the many nonstandard implementations that attempt to address Pascal's widely acknowledged limitations in such areas as string manipulation and file handling.

Given its narrow focus on the needs of experienced programmers and its strict adherence to the ISO standard, *Pascal for Programmers* serves its intended audience superbly. The first chapter offers a brief introduction and even briefer history of Pascal, followed by practical suggestions on how to use the book. Chapters 2 through 5 present the basic building blocks that constitute Pascal programs. The remaining ten

chapters present the different structures that enable the programmer to construct complete, complex programs from these basic building blocks.

Each chapter deals with one particular programming construct, relating to either data structures or the implementation of algorithms. Although each chapter is relatively independent of the others, they are presented in an orderly sequence that introduces the most difficult concepts toward the end. For the programmer already familiar with Pascal, this facilitates quick reference to a complete discussion of a particular construct, without the need to leaf through the entire book.

Examples are used liberally throughout. These range from simple, short procedures to complete, complex, and lengthy programs, depending on the topic under discussion. In this regard, the text is far superior to others that use only a syntax chart and a statement or two to illustrate a point. While such an approach may focus the reader's attention on the construct under consideration, it does not offer the broader view used throughout *Pascal for Programmers*. The examples are generally followed by remarks that amplify and add to the sense of the code itself. As the book progresses, both the examples and the accompanying remarks become longer and more complex, as Pascal's power and elegance are more fully demonstrated.

Exercises are included as well, but they are not the true/false or short-answer variety commonly found in more introductory tests. Rather they require complete and challenging programs or program fragments. An appendix offers solutions to one exercise selected from each chapter. Other appendices include collected syntax diagrams, a bibliography, tables and reference lists, and a discussion of implementation-defined and implementation-dependent features.

More and more programmers and educators are coming to believe that, once one understands the fundamental principles of programming, any given language represents only a particular embodiment of that relatively fixed set of basic constructs. Conversely, if the student lacks an integrated view of programming, each new language will pose a seemingly new challenge, unrelated to

other languages learned previously.

Pascal for Programmers seems written with the broad view in mind. While it does not directly contrast Pascal code with that of other languages, high-level concepts are discussed from the perspective of language independence before the particulars of their embodiment in Pascal are introduced.

While the text is inappropriate to the needs, background, or budget of the curious beginner, its thorough, practical focus on the essence of Pascal qualifies it as a worthwhile continuing-education and reference tool for the programmer. If you want to teach yourself Pascal in a few evening's time or brush up on its more complex features without wading through background or introductory material of secondary importance, *Pascal for Programmers* is perhaps the best such text available today.

Microcomputer Software Design

by Sally Campbell

Published by Prentice-Hall

\$12.95

ISBN: 0-13-580621-6

Reviewed by Richard L. Lozes

First, a disclaimer: Microcomputer software design is no different than any other software design. However, because microcomputers are also consumer items, the title appeals to a mass market. How well does the book address its topic in that market?

As the author admits in her preface, professional programmers write and study texts of greater breadth and depth. Indeed, an example she chose to develop throughout the book is amusing. I doubt that anyone with only the study of this book could cost effectively produce a correct payroll system. Such software is much better purchased.

Ms. Campbell surveys many of the techniques used in commercial software creation, all of which were invented in the sixties and seventies. Hierarchical Input-Process-Output (HIPO) charts, pseudocode, flowcharts, modularization, and testing are all discussed.

The first chapter could well have been left out. It is no more than a museum catalog of computer parts, imparting no understanding of software

design. (For instance, whether a computer has a motherboard really has no bearing on how it is programmed.) The second chapter, on problem definition, glosses over the specification phase in its haste to begin design work. Such haste is well known to be fatal to programming efforts.

The author ignores two of the overriding characteristics of software creation as it is currently practiced: programming is much more of a social activity, and programming is very expensive. The expense is the reason for the existence of the techniques discussed. And, true, individual programmers *do* use techniques, but they also participate in reviews, walk-throughs, and heated discussions. One would like to see some acknowledgement of this "engineer's" viewpoint as motivation.

Microcomputer Software Design can be recommended only to those with some knowledge of the programming process who desire a shallow introduction to HIPO, pseudocode, file definitions, or debugging. Otherwise, \$12.95 can be better spent elsewhere.

Self-Organization and Associative Memory

by Teuvo Kohonen

Published by Springer-Verlag

ISBN: 0-387-12165-X

Reviewed by Richard L. Lozes

Associative memory is a behavior peculiar to certain adaptive filters. If x denotes a key, M a set of (memory) parameters, and y data to be recalled, then

$$y = y(x, M)$$

represents associative recall if $M = M$ (historical y).

Professor Kohonen develops his book around this type of function, seeking the conditions under which memory is able to store compressed data and to reexpand it for recall. He pursues notions of input-driven system reorganization, devoting a chapter to "self-organizing feature maps."

Many of us in the computer field are familiar with content-addressable memory. Such a memory is characterized by key-data pairs that are stored as received, one pair per location. Recall is effected by specification of a key alone and involves only the word(s)

holding the requested key. This simple, direct, one-to-one mapping is a good deal less general than the associations that Professor Kohonen discusses.

Associative memories are characterized by a distribution of data throughout the memory medium, similar to the way in which a hologram stores data, but they also possess key-based retrieval the way content-addressable memories do. Mathematically, this distributed memory is reflected in the changing of the entire set of parameters, M —thus, the notion of a system reorganization. This is the type of memory that typifies the book. Clearly, the author is really attempting to answer the enigma of biological memory in preference to the well-understood technological memories.

The second chapter surveys and summarizes the necessary mathematical tools. These are drawn almost entirely from the domain of linear mathematics—in particular, from linear algebra. Nonlinearity, not being required at the lowest levels (within each memory element, say), is deemphasized. Actually, feedback via the associative parameters, M , above suffices to introduce the memory behavior. This chapter is particularly easy to read, despite the breadth of its coverage. In good European tradition, the author has amply referenced the original literature.

I must level one serious criticism against this monograph. Professor Kohonen refuses to permit arbitrarily complex memory elements. Rather he insists that they be "physical." I believe that he is mistakenly assuming that "physical" systems are inherently and economically describable by simple mathematics. That strikes me as naive. On the one hand, one can always reduce complex systems to simpler components, either exactly or approximately. On the other hand, nothing indicates that biological memory is characterized by (mathematically) simple cells. In short, while Occam's razor demands simplicity, Professor Kohonen apparently has shaven away far more than fuzziness.

Nevertheless, as an introduction to the field of associative memory, this little book is quite reasonable. A reader of modest mathematical prowess will receive sufficient inspiration and

information here to start into the field.

Interactive Programming Environments

*Edited by David R. Barstow,
Howard E. Shrobe, and
Erik Sandewall*

Published by McGraw-Hill

\$34.95, 609 pages

Reviewed by Tom Provenzano

A great deal of research is in progress in the area of automated software engineering environments—what some might call expert systems for programmers. Software engineering environments provide an integrated set of tools to assist the programmer during most phases of the software development process, from system design to software testing. *Interactive Programming Environments*, edited by Barstow, Shrobe, and Sandewall, is a collection of 28 papers describing the research under way and the philosophy behind the building of programming environments.

The book is divided into five sections that explore issues and problems in the construction of interactive programming environments (IPEs). The editors admit in the preface that “programming environment” does not have a generally accepted meaning, but they offer a definition of “computer-aided design systems for software.”

The papers in the book focus on interactive as opposed to batch types of environments, because the writers feel that a dialogue form of interaction between man and machine is the optimal method of implementing and using software engineering environments. None of the papers addresses automated software production—programs that write programs—since that is an entirely separate field of its own, heavily submerged in artificial intelligence research. What is addressed in these papers is the middle ground between manual systems (the way software is currently developed) and automated systems, namely IPEs.

The first section in the book, “Perspectives on Interactive Programming Environments,” consists of three papers exploring the motivations and reasons for developing IPEs. The first paper, “Breaking the Complexity Barrier (Again),” by Terry Winograd of Stanford University, written in 1973,

sketches what an IPE should be like and what it must do to help programmers handle the complexity involved in building modern software systems. Winograd describes an integrated system of tools containing knowledge about the system being built and acting as a sort of “programmer’s assistant” by providing precompilation error checking, system question answering, and debugging support throughout all stages of the software development process. Most of the other papers in the book describe systems that attempt to implement what Winograd outlines in this early paper.

One major issue that becomes apparent after reading the papers in this book is that of the difference between the incremental growth vs. the life cycle approach to software development. Incremental growth is akin to the software prototyping approach of building a working base and adding enhancements piece by piece, whereas the life cycle method follows a more rigid and inflexible structure of design/code/test. Because most of the papers are authored by people involved in academic or research efforts, the former approach seems to be preferred. A question to keep in mind when reading these papers is: Is the incremental growth method of software development necessarily superior to the life cycle method, especially for systems built in a nonacademic, industrial environment?

Other IPE features discussed throughout the book include data-driven systems, smart editors and debuggers, and DWIM (“Do What I Mean, Not What I Say”) facilities. A primitive example of the last-named feature is having the system overlook simple typing errors and perform the command closest to what the user intended (after querying to ensure it’s OK to do so).

Included in this volume are papers describing programming environments based on LISP (INTERLISP, LISP Machine, the Programmer’s Assistant), C (Unix, Programmer’s Workbench), Pascal (Pathcal), Ada (APSE—Ada Programming Support Environment), and smart editors (EMACS, DED, MENTOR). The book’s final section, “The Future of Interactive Programming Environments,” contains another paper by Terry Winograd in which he

conjectures that the next step in IPEs must be the elimination of the necessity of having programmers work with specific programming languages. Instead they will be developing software at a level conceptually above high-order languages. Systems will be built out of software components, and the programming environments used to build such systems will know a great deal about software and software development in general.

Interactive Programming Environments is a good source of information on the continuing research in a field that shows promise in improving and altering the way software systems are developed. The book is fairly easy reading (a knowledge of LISP is helpful but not essential for some of the papers) for those with software engineering backgrounds and will be much appreciated by those implementing or interested in automating the software production process.

I would hope, though, that all implementors of future IPEs might take a hint from the Emily system described in the book and provide a “sympathy” button, as explained by Wilfred J. Hansen: “When the user is frustrated he can push a sympathy button. In response, Emily displays at random one of ten sympathetic messages.” We can all use a pat on the shoulder from our digital friends now and then.

Talking Chips: IC Speech Synthesis

by Nelson Morgan

Published by McGraw-Hill

178 pages, 150 illustrations,

\$24.50

Reviewed by Dennis Cashton

One of the fastest growing areas of technology for computers is speech synthesis. But it is not a new area. Work on this electronic marvel has been going on since 1939 when a device called VODER was introduced. If this subject holds the least bit of interest for you, this work is an excellent addition to your library.

At first glance, you might think that unless you are an engineer this book is too technical. Upon further examination, however, you would find that it is entertaining and not at all intimidating. To be sure, this is not a “how to” book but more of an explanation of

and I enjoyed reading it.

Nelson Morgan has written a serious book, but this has not stopped him from making it very amusing. He makes occasional references to some very silly uses for speech synthesis, the first of which is his mythical "talking toaster." There are numerous cartoons depicting equally silly situations with speaking devices. Even with all the frivolity, he never fails to stay on target with his writing. He starts out with a description of what speech synthesis really is then gives a basic overview of the several ways it can be accomplished. Continuing on through the hardware, he gets to a description of how some of today's speech synthesis chips work.

The next part of the book is where things get a little deep for the nontechnical person. This is where Mr. Morgan gets into the actual analysis of speech, including waveforms, estimation, classification, and pitch tracking (to drop a few names). He then goes on to discuss synthesis by rule and spends an entire chapter on the subject of obtaining quality audio to use for creating synthesized speech.

The final chapter covers how to send your own speech to an IC manufacturer and have it put on a chip, new ideas for speech synthesis, music and sound effects, and a section on speech recognition. After reading this last section you will understand why a lot more work has to be done in this area.

On the whole, the author does a wonderful job of presenting his topic. He compares the different techniques for speech analysis and synthesis, and he points out why some companies are using some methods and others are using different ones. If waveforms are not your thing, but you are interested in the concept of speech synthesis, it is quite simple to skip right over the more technical sections and still get a lot out of the book. But if you like formulas and models and such, the book is filled with them. There's even an appendix that describes Fourier analysis and windowing in enough detail to delight any calculus fan.

For a subject as complex as speech synthesis, this is a clear and pleasant presentation. I learned a lot from it.

The Acorn/BBC Micro, An Expert Guide

by Mike James

Published by Prentice-Hall, Inc.

\$14.95, 158 pages

Reviewed by Morton F. Kaplon

Who better than the author to tell you what this book is all about? The first paragraph of the Preface summarizes it quite nicely.

“The subject of this book is the BBC Micro, its hardware, and its software, and it is aimed at anyone who has already started to plumb the depths of this fascinating machine. It is not an introduction to BASIC, nor does it attempt to explain the fundamental hardware that goes to make up any computer. Instead it plunges straight into the complexities and intricacies of this very special micro.”

The book is divided into nine chapters of approximately equal length, entitled: The Hardware; BBC BASIC; The Machine Operating System; The Video Display; The Sound Generator; Interfacing; Introduction to Assembly Language; Assembly Language II; and Postscript. The author notes that material covered in the *BBC User Guide* is not duplicated in this volume except as required to make this book self-contained.

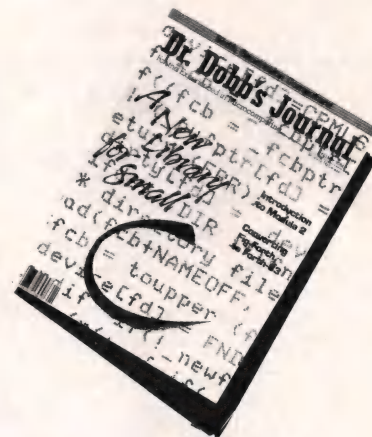
The BBC Micro is based on the 6502 microprocessor operating at a clock speed of 2 MHz with a maximum of 32K of RAM. The standard system has 32K of ROM in the form of two 16K chips. One contains the operating system (MOS) and 1K used principally for memory-mapped I/O devices. The second chip contains the BBC BASIC interpreter and the 6502 assembler; this ROM can be replaced by any of three alternative ROMs under software control. Storage is by tape cassette.

The book does as it says. It deals with this machine at the level stated and with those assumptions. If you own a BBC Micro and want to know more about it, then this book may prove useful. If you are not a BBC Micro owner, then it is difficult to see what this book may hold for you or how it could be useful to you. This is not written as a "cook book," and the implementation of the concepts presented will require serious dedication

Advertisers!

Get Ready For December

Dr. Dobb's Journal

special **UNIX** issue

Space Reservation Deadline

OCTOBER 12, '84

Materials Deadline

OCTOBER 19, '84

Contact:

Walter Andrzejewski

Alice Hinton

(415) 424-0600

Dr. Dobb's Journal
2464 Embarcadero Way
Palo Alto, CA 94303
(415) 424-0600

on the part of the user.

New Books

Personal Pascal

David E. Cortesi and

George W. Cherry

Reston Publishing Co., Inc.

Reston, VA 1984

420 pages

This book is the wedding of George Cherry's *Pascal Programming Structures* (Reston, 1981) with the IBM PC. Our own resident intern, Reverend D. E. Cortesi, officiates.

microSUB:MATH

A library of **Numerical Methods Subroutines** for use with your **FORTRAN** programs.

Over sixty subroutines of:

FUNCTIONS
INTEGRATION
MATRICES
NON-LINEAR SYSTEMS

INTERPOLATION
LINEAR SYSTEMS
POLYNOMIALS
DIFFERENTIAL EQ

Versions now available for:
MS-DOS: IBM FORTRAN-77
SuperSoft FORTRAN IV
Microsoft MS FORTRAN ver 3.2
DRI DR FORTRAN-77
CPM-80: Microsoft F-80
SuperSoft FORTRAN IV

LICENSE, \$250.
with SOURCE CODE, \$600.
(Manual alone, \$25.)

MATH
SUBROUTINE
LIBRARY

TRADEMARKS
Microsoft & MS
CPM & DR FORTRAN-77
SuperSoft
Digital Research Corp
Interpretation Business Machines
IBM
SuperSoft Corp.

Circle no. 28 on reader service card.



foehn consulting, PO Box 5123, Klamath Falls, OR 97601 503/884-3023

Microprogrammer's Market 1984

Marshall Hamilton

TAB BOOKS Inc.

Blue Ridge Summit, PA, 1984

\$13.50, 229 pages

You can find interesting information in this book even if you're not looking for a job. For example: prior to this year, Data General published fewer than 50 programs (ever), but the company plans to sell 200 in 1984. Yeah, Data General. The computer company with soul.

The Best of CP/M Software

John D. Halamka

SYBEX Inc.

Berkeley, CA 1984

\$14.95, 252 pages

One man's opinion about 45 software packages. Although no reader will agree with all of his choices, Halamka has much software reviewing experience and writes well. One weakness, at least from a programmer's viewpoint, is the weak treatment of languages and utilities.

Macintosh! COMPLETE

Doug Clapp

Softalk BOOKS

North Hollywood, CA 1984

\$19.95, 329 pages

As of this writing, one of two decent books on the Mac.

The Apple Macintosh Book

Cary Lu

Microsoft Press

Bellevue, Washington, 1984

\$18.95, 383 pages

As of this writing, one of two decent books on the Mac. This is the technical one.

The Small-C Handbook

James E. Hendrix

Reston Publishing Co., Inc.

Reston, VA 1984

\$14.95, 256 pages

DDJ published Ron Cain's small compiler for a subset of the C language in the May 1980 issue. By the time (December 1982 - January 1983) DDJ published Jim Hendrix's greatly-enhanced Small-C, the issues containing Cain's original compiler and subsequent issues containing Small-C mate-

rial had sold out. The Hendrix issues soon sold out as well. This book describes the latest version of the compiler, Small-C Version 2.1.

A Programmer's Notebook:

Utilities for CP/M-80

David E. Cortesi

Reston Publishing Co., Inc.

Reston, VA 1983

\$16.95, 368 pages

You won't see this book reviewed here because of our policy of not reviewing our editors' books. We will, however, read it. Cortesi believes that programming is neither an art nor a science, but a craft. This book is about learning the craft.

The Encyclopedia of Microcomputer Terminology

A Sourcebook for Business

and Professional People

Linda Gail Christie and

John Christie

Prentice-Hall

Englewood Cliffs, NJ, 1984

\$9.95,

We've needed a good dictionary/encyclopedia of microcomputer terminology for a long time. We still do.

Apple Software Directory

Yellow Pages to the World of

MicroComputers

PC Telemart/Vanloves


PC Telemart, Inc./Vanloves

Fairfax, VA, 1984

\$24.95, 965 pages

The publisher bills this book as the yellow pages to the world of microcomputers, and it does have the style and heft of a metropolitan phone directory. For this third edition, its compilers have added a mediocre glossary and a list of bulletin boards, and have updated its information on software for the Apple II-III family. Scads of software.

DDJ



CP/M KEYBOARD MACRO EXPANDER
POWERFUL NEW UPGRADE!
ALSO AVAILABLE FOR CROMEMCO CDOS

PRO microSystems
16609 Sagewood Lane
Poway, California 92064
(619) 693-1022

Dealer and OEM inquiries welcomed

Set up at
The National Software Show
Alhambra Convention Center
September 5-7, 1984
Booth No. 2404

CP/M Im Digital Research, Inc.

Circle no. 55 on reader service card.

CROSS-DEVELOPMENT SOFTWARE TOOLS

C COMPILERS (with ROM support)

host IBM/PC, target 6809
PDP-11,
6809

MACRO ASSEMBLERS

host IBM/PC, target 6801, 6805
PDP-11, 68HC11, 6809,
6809 16000, 68000

IBM/PC: TM of Int'l Business Machines.
PDP-11: TM of Digital Equipment Corp.

INTROL CORPORATION
647 W. Virginia St.
Milwaukee, WI 53204
(414) 276-2937

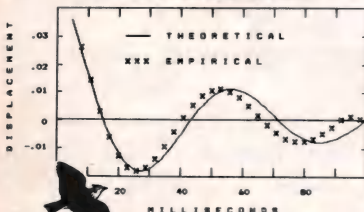
Circle no. 36 on reader service card.

Graphs without Graphics?

No need for screen graphics. Publishable graphs on your dot matrix printer.

Easy to Use. No programming.
CP/M 80 or 86, MS-DOS, or PC-DOS.
Excellent Manual. Most disk formats.

DataPlotter™



Lark Software™
7 Cedars Road
Caldwell, NJ 07006

Line Graphs & Scatterplots...\$69
Bar Graphs & Pie Charts...\$69
Both for...\$99
(Prices include manual)

Shipping.....add \$3
Outside US and Canada.....add \$6
Specify which Printer Via, M/C
(201) 226-7552

Circle no. 42 on reader service card.

TEK-MAR

HIGH-LEVEL FORTRAN GRAPHICS LIBRARY FOR THE IBM PC

Features:

- Windowing • Viewporting • Clipping
- Axis Rotation • Screen Dump & Restore
- Dump Screen Graphics to Epson

INCLUDES EXAMPLE APPLICATION SOURCE CODE

REQUIRES:

- 320K Memory • Two DS Disk Drives
- TecMar Graphics Master Board
- MS Fortran 3.20 or higher
- Optional Plotter (Western Graphic 4636)

\$195

ADVANCED SYSTEMS CONSULTANTS

18653 Ventura Boulevard, Suite 351
Tarzana, California 91356
(818) 990-4942

Circle no. 2 on reader service card.

NOW C HERE! CROSS SOFTWARE for the NS32000

Also Available for IBM PC
INCLUDES:

- Cross Assembler •
- Cross Linker •
- Debugger •
- N.S. ISE Support •
- Librarian •
- Pascal Cross Compiler •
- C Cross Compiler •

U.S. prices start at \$500

SOLUTIONWARE

1283 Mt. View-Alviso Rd.
Suite B
Sunnyvale, Calif. 94089
408/745-7818 • TLX 4984264

Circle no. 68 on reader service card.

ICs PROMPT DELIVERY!!! SAME DAY SHIPPING (USUALLY)

DYNAMIC RAM

256K	256Kx1	150 ns	\$34.34
256K	256Kx1	200 ns	26.26
64K	64Kx1	150 ns	4.77
64K	64Kx1	200 ns	4.62
16K	16Kx1	200 ns	1.21

EPROM

27256	32Kx8	300 ns	\$49.97
27128	16Kx8	300 ns	18.77
27C64	8Kx8	200 ns	22.50
2764	8Kx8	250 ns	6.50
2732	4Kx8	250 ns	6.37
2716	2Kx8	450 ns	3.50

STATIC RAM

6264LP-15	150 ns	\$31.25
6116P-3	150 ns	6.36

Open 6 1/2 days. We can ship via Fed-Ex on Sat

MasterCard/VISA or UPS CASH COD

Factory New, Prime Parts JUP
MICROPROCESSORS UNLIMITED
24 000 South Peoria Ave. (918) 267-4961
BEGGS, OK 74421

Prices shown above are for September 17, 1984
Please call for current prices & volume discount. Prices subject to change. Please expect higher prices on some parts due to world wide shortages. Shipping and insurance extra. Cash discounts on some items. Small orders received by 6 PM CST can usually be delivered to you by the next morning, via Federal Express Standard Air @ \$5.99

Circle no. 48 on reader service card.

C Programmers B-Trees

For
\$75.00 + 2.00 Postage

Source Code Included

The Softfocus B-Trees record indexing library will help you develop sophisticated application programs.

With Softfocus B-Trees you get:

- high speed file handling for up to 16.7 million records per file
- customizable BDS or K & R standard C source code
- no royalties on applications programs
- support random and sequential file access
- includes example programs

Softfocus

1277 Pallatine Dr., Oakville, Ont.
Canada L6H 1Z1 (416) 844-2610

Circle no. 65 on reader service card.

Get the power of your Z80, and the elegance of direct access to CP/M functions from your high level programs with

SYNLIB utility library

SYNLIB consists of MICROSOFT compatible object code that may be called from any high level language that uses MICROSOFT parameter passing conventions.

SYNLIB gives you extremely powerful array and buffer manipulation using the Z80 LDIR instruction; program access to the CP/M CCP console command line; high speed disk block I/O; a high quality random number generator; hex to ASCII conversion optimized by special Z80 instructions; program chaining; and more.

And, because our programmer abhors a vacuum, each 8" floppy comes packed with some of the most valuable public domain software, including available source, absolutely free. You get **SWEEP**, a menued disk utility that makes a computer phobe a systems programmer; **UNSPLOOL**, so you can print and use your computer without buying an expensive buffer; **I**, to get multiple commands on a line; **MODEM7**, so that you too can join the free software movement; and many others.

SYNLIB \$50.00 8" SSD CP/M format
SOURCE: \$100.00

Licensing for commercial use available.

SYNTAX CONSTRUCTS, Inc.
14522 Hiram Clarke, Houston, Texas 77045
(713) 434-2098

CP/M is a registered trademark of Digital Research, Inc.
Microsoft is a registered trademark of Microsoft Corp.

Circle no. 70 on reader service card.

A general purpose programming language for string and list processing and all forms of non-numerical computation.

SNOBOL4+ —the entire

SNOBOL4 language with its superb pattern-matching facilities • Strings over 32,000 bytes in length • Integer and floating point using 8087 or supplied emulator

- ASCII, binary, sequential, and random-access I/O • Assembly Language interface • Compile new code during program execution • Create SAVE files • Program and data space up to 300K bytes RAM

With **ELIZA** & over 100 sample programs and functions

For all 8086/88 PC/MS-DOS or CP/M-86 systems, 128K minimum 5 1/4" DSD, specify DOS/CPM format

Send check, VISA, M/C to:

Catspaw, Inc. plus \$3 s/h
P.O. Box 1123 • Salida, CO 81201 • 303/539-3884

Circle no. 9 on reader service card.

4 = 20 ?

4 the VIC!

VIC FORTH® for the VIC 20™

VIC FORTH® cartridge \$14.25
memory expansion optional

Starting FORTH by L. Brodie \$18.50
softcover book/recommended

BOTH TOGETHER (mention this ad) \$30.75

PA residents add 6%

Please request our free VIC brochure



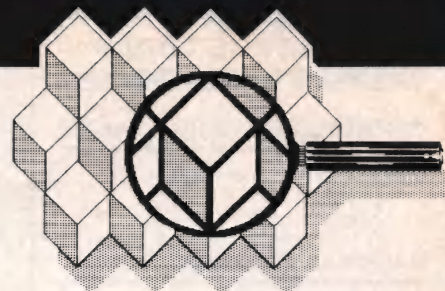
EMGE ASSOCIATES

P.O. Box 17330

Pittsburgh, PA 15235-0330

VIC FORTH® HES VIC™ Commodore

Circle no. 25 on reader service card.



by R. P. Sutherland

Languages

Starlight Forth Systems has a Forth compiler with macro assembler and debugging tools for the new 16-bit 65SC802 and 65SC816 microprocessors (which are plug and software compatible with the 6502 and 65C02). Options include a full assembler for all versions of the 65xxx processor family, a meta-compiler with code and high-level debugging tools, source on disk, and a full-screen editor. High-level listings of the tools for non-6502 systems are also available. For the full range of options and prices write: Starlight Forth Systems, 15247 N. 35th St., Phoenix, AZ 85032. **Reader Service No. 105.**

Wizard Systems Software has released version 1.3 of the **Wizard C Compiler**. Improvements over the previous version, which already supported the full C language, include improved syntax checking and error diagnosis of the compiler, warnings about mistaken use of the assignment operator where the equal operator was intended (mis-typing = for ==), as well as indications where parentheses may be needed. Another version 1.3 enhancement is the support of the Large Memory Model, which allows Wizard C users the capability of writing programs as large as all of physical memory, without restriction. Wizard C Compiler is available for \$450.00 (single CPU license) from Wizard Systems Software, Inc., 11 Willow Court, Arlington, MA 02174 (617) 641-2379. **Reader Service No. 107.**

C-Power Packs feature six libraries of functions and subassemblies to aid in debugging and to enhance code uniformity. C-Power Packs operate on Microsoft and Lattice C Compilers and include code for data base, communications, mathematics, utilities, and building block functions. Prices

start at \$99.00. For more information contact Dr. Ramal Murali, Software Horizons, 165 Bedford Street, Burlington, MA 01803 (617) 273-4711.

Reader Service No. 109.

Macintosh

MacTools is a low-cost disk utility package that integrates many standard disk functions into one program. In addition to common disk manipulations such as copying, renaming, and deleting files and copying, renaming, and formatting disks, MacTools provides features such as: (1) copy protect/unprotect; (2) verify a disk; (3) lock/unlock files; and (4) make files visible/invisible. MacTools is available for \$39.95 from Central Point Software, Inc., 9700 SW Capitol Hwy, Suite 100, Portland, OR 97219 (503) 244-5782. **Reader Service No. 111.**

SoftTech Microsystems is shipping three packages for Macintosh programmers. **UCSD Pascal Development System** allows access to mouse, graphics, and text fonts provided by Mac ROM routines for \$195.00. **Fortran-77 Development System** provides an ANSI-77 subset Fortran for \$295.00. The **Advanced Development Tool Kit** includes source code for graphics/mouse interface, a symbolic debugger, 68000 assembler, and a linker. The cost is \$150.00. The company claims that UCSD Pascal and Apple Pascal are sufficiently compatible so as to allow porting of Apple II and III programs to the Macintosh. In addition, the company says that UCSD applications running on the IBM PC can be easily ported. SoftTech Microsystems is located at 16885 West Bernardo Drive, San Diego, CA 92127 (619) 451-1230. **Reader Service No. 113.**

The Programmer's Shop has established the **Macintosh Developer's Ex-**

change, a bulletin board to provide a forum for discussions related to Macintosh development. They hope to facilitate a freer flow of information and to promote the exchange of small utilities of the type usually written for in-house use. Some intriguing products are in the works that will help convert assembly language programs written for the IBM PC, the 8088 CP/M environment, and Apple IIe to the Macintosh. To participate in the Macintosh Developer's Exchange contact the Programmer's Shop, 128 Rockland St., Hanover, MA 02339 (800) 421-8006 or use the bulletin board (P-line) at (617) 826-4086.

Expert Systems

Expert-Ease is an expert system generator that runs on a microcomputer. The product is based on the artificial intelligence "inference engine" developed by Intelligent Terminals, Ltd., in Edinburgh, Scotland. An expert gives examples to the system, and once Expert-Ease has enough examples, it figures out the rules that lead to the expert's conclusions. Then the program coaches the expert to enter questions, which later will aid the nonexpert in providing information for solving new problems. Expert-Ease runs on the UCSD-p system on the the IBM PC and compatibles. Separate versions are available for DEC Rainbow and Victor 9000 (or Sirius). The full software package costs \$2,000.00. A demonstration diskette and manual are available for \$125.00. Contact Expert-Ease Inc., 206 Fifth Avenue, New York, NY 10010 (212) 684-4331. **Reader Service No. 101.**

The Forth Interest Group (FIG) invites you to attend the **Forth Modification Laboratory (FORML)** conference at the Asilomar conference grounds in

Pacific Grove, California, from November 23 – 25. Expert systems and artificial intelligence are among the topics that will be covered. For more information call the FIG hot line (415) 962-8653.

Jack Park's **Expert-2** (see *DDJ*, April 1984) is now optimized for use with MMSFORTH versions 2.0 and up. MMSFORTH licensees can add Expert-2 to their system for \$69.95. The MMSFORTH system disk is available for TRS-80 Models 1, 3, 4, and 4P for \$129.95. **Reader Service No. 103.**

Disk Storage

Chipmunk is a 3.5-inch portable disk drive for the TRS-80 Model 100 and other lap computers. The Chipmunk is powered by internal rechargeable nickel cadmium batteries or an AC

adaptor. The unit plugs into the computer's expansion bus using a 1-inch wide ribbon cable and EZ-Snap connector. The Chipmunk Disk Operating System, CDOS, is contained on a ROM chip. The price is \$499.00. For more information write Holmes Engineering, 5175 Greenpine Drive, Salt Lake City, UT 84123 (CompuServe 71675,527). **Reader Service No. 115.**

Miscellany

Free-Access On-Line Software Library

A Florida firm specializing in computer-readable data bases has opened an electronic software locator service to assist software shoppers in finding software products. Anyone with a telephone and a modem-equipped computer (set for 8 bits, 1 stop bit, no parity)

can log on by calling (305) 845-6466. I found useful categories and thousands of software products with descriptions and prices. Shoppers can enter information, comments, or requests directly into a software vendor's electronic mail box. For more information, direct inquiries to Searchmart Corporation, 745 U.S. Highway One, North Palm Beach, FL 33408 or call (305) 845-2996.

Free Apple Software

Send one dollar to Computer Learning Center, P.O. Box 45202, Tacoma, WA 98445 for your catalog of public domain software for Apple II+ (and some Apple IIe). *Forbidden Fruit* contains thousands of programs of every possible variety, which you can order for the price of the floppy (\$4.00).

CP/M Utilities Catalog

John Donohue has compiled an interesting catalog of low-cost CP/M utilities. A sample entry called "Later, listed for \$22.00, allows one to accumulate lists of commands that can be postponed. According to the catalog: "When you go to lunch, type SUBMIT NOW, and they all get done. Let the computer remember what it has to do, and do it when you're not around. Obvious." None of the utilities are over \$50.00 and some are as low as \$7.00. Order your copy from Donohue & Co. Computer Services, P.O. Box 255, Hannibal, NY 13074. **Reader Service No. 119.**

IEEE Call for Papers

The deadline for submitting papers to the IEEE Computer Society Conference on Computer Vision and Pattern Recognition is January 7, 1985. For a copy of the call for papers with a complete list of conference topics and full details on submission of papers, write or call: Computer Vision and Pattern Recognition, P.O. Box 639, Silver Spring, MD 20901 (301) 589-8142. The conference will be held in San Francisco at the Cathedral Hill Hotel from June 9 – 13, 1985. **Reader Service No. 121.**

DDJ

Reader Ballot

Vote for your favorite feature/article.
Circle Reader Service No. 200.



Tera-drive is a high-capacity floppy disk subsystem for Apple IIe and II+. Using cobalt-coated, high-density, 5¼-inch floppy disks, this unit provides one megabyte of storage per disk. The proprietary operating system supports UCSD-p, CP/M, and DOS. The 1-

Mbyte drive retails for \$995.00 and the 2-Mbyte dual drive retails for \$1595.00. For further information contact Eicon Research, Inc., 520 Fifth Avenue PH, New York, NY 10036 (212) 719-5353. **Reader Service No. 117.**

C

Software Development

PCDOS/MSDOS

Complete C Compiler

- Full C per K&R
- Inline 8087 or Assembler Floating Point, Auto Select of 8087
- Full 1Mb Addressing for Code or Data
- Transcendental Functions
- ROMable Code
- Register Variables
- Supports Inline Assembler Code

MSDOS 1.1/2.0

Library Support

- All functions from K&R
- All DOS 2.0 Functions
- Auto Select of 1.1 or 2.0
- Program Chaining Using Exec
- Environment Available to Main

c-window™

Symbolic Debugger

- Source Code Display
- Variable Display & Alteration Using C Expressions
- Automatic Commands
- Multiple Breakpoints by Function & Line Number

8088/8086 Assembler

- FAST — Up to 4 times Faster than IBM Assembler
- Standard Intel Mnemonics
- Compatible with MSDOS Linker
- Supports Full Memory Model

8088 Software Development Package

\$199⁰⁰

Includes: C Compiler/Library, c-window, and Assembler, plus Source Code for c-systems Print Utility

c-systems

P.O. Box 3253
Fullerton, CA 92634
714-637-5362

Circle no. 16 on reader service card.

ADVERTISER INDEX

Reader Service No.	Advertiser	Page No.	Reader Service No.	Advertiser	Page No.
1	ABComputing	109	40	Laboratory Microsystems	117
2	Advanced Systems Consultants	125	41	Lahey Computer Systems	79
3	Ampro Computers, Inc.	105	42	Lark Software	125
4	Avocet Systems, Inc.	17	43	Lattice, Inc.	79
5	BD Software, Inc.	57	44	Leo Electronics, Inc.	105
6	B.G. Micro	25	45	Lifeboat Associates	64 - 65
7	Borland International	OBC	46	MicroMotion	79
8	California Digital Engineering	103	47	Microprocessors Engineering	73
9	Catspaw	125	48	Microprocessors Unlimited	125
10	The Code Works	73	49	OPT-Tech Data Processing	97
11	Computer Friends	59	50	Phoenix Computer Products	9
12	Computer Innovations	113	51	Poor Person Software	103
13	Computer Innovations	115	52	Procode International	59
14	Computing	95	53	Programmers Connection	83
15	Creative Solutions	16	54	The Programmer's Shop	47
16	C Systems	128	55	Pro Microsystems	124
17	C User's Group	86	56	QCAD	57
18	C Ware	107	89	QUAID Software LTD	107
83	DDJ Bound Volume	87	57	Quest Research	IBC
84	DDJ Back Issues	107	58	Rational Systems, Inc.	97
85	DDJ Subscription Problem	109	*	Edward Ream	55
86	DDJ Subscription	115	60	Redding Group	113
87	DDJ Reprints	117	61	Revasco	55
88	DDJ Advertising	123	62	SemiDisk Systems	97
19	Data Access Corporation	15	63	Simpliway Products Company	86
20	Data Base Decisions	21	64	SLR Systems	41
21	Datalight	69	65	Softfocus	125
22	Dedicated Microsystems	55	66	The Software Bottling Company	IFC
23	D + V Systems	83	82	Software Horizons, Inc.	29
24	Ecosoft, Inc.	83	67	Solution Systems	49
25	EMGE Associates	125	80	Solution Systems	47
26	Essential Software	41	81	Solution Systems	47
27	Faircom	73	68	Solutionware	125
28	Foehn Consulting	124	69	Starlight Forth Systems	113
29	GGM Systems, Inc.	43	70	Syntax Constructs	125
30	Greenleaf Software Inc.	43	71	Systems Engineering Tools	27
31	GTEK	69	72	Thunder Software	128
32	Hallock Systems Consultants	13	73	Unified Software Systems	69
33	Harvard Softworks	59	74	Vance Info Systems	46
*	IBM	2 - 3	75	Visual Age	118
35	Integral Quality	99	76	Mark Williams Company	7
36	Introl Corporation	125	77	Wizard Systems	41
37	IQ Software	1	78	WL Computer Systems	35
38	Key Solutions, Inc.	35	79	Workman & Associates	46
39	Korsmeyer Electronics Design	35			

Thunder Software

- **The THUNDER C Compiler** - Operates under the APPLE Pascal 1.1 operating system. Create fast native 6502 programs to run as stand alone programs or as subroutines to Pascal programs. A major subset of the C defined by K & R. Includes a 24 page users guide, newsletters, Macro preprocessor, runs on APPLE II+, IIe, IIc. Source code for libraries is included. **Only \$49.95**
- **ASSYST: The Assembler System** - A complete 6502 editor/assembler and linker for APPLE DOS 3.3. Menu driven, excellent error trapping, 24 p. users guide, demo programs, source code for all programs. Great for beginners. **Only \$23.50**
- **THUNDERXREF** - A cross reference utility for APPLE Pascal 1.1. XREF generates cross references for each procedure. Source code and documentation provided. **Only \$19.95**

Thunder Software POB 31501 Houston Tx 77231 713-728-5501
Include \$3.00 shipping. COD. VISA and MASTERCARD accepted

Circle no. 72 on reader service card.

We thought about calling it MacSimplex . . . after all it makes your IBM® PC behave like a Macintosh™ and much more . . .

and with over two years in the making, the Simplex Database Management System has features like 32-megabyte virtual memory and the most powerful networked/relational database in the microcomputer industry. Simplex was designed around how you think and the Macintosh way, so that you can use your favorite mouse to handle those mundane tasks like menu selection and data manipulation. And, if you don't have a mouse, you can use our keyboard mouse simulator, MouSim™.

Pop-up and pull-down menus, dialog and alert boxes are not just added features, they are the heart of the Simplex way. In addition, Simplex gives you both a software and a hardware floating point capability, each with 19-digit accuracy. It permits login, password, privilege, and can be used on a local area network. Simplex has full communications and a remote or local printer spooler. Above all, Simplex is modular and grows with you! Simplex also has a full-featured, English-like language which is simple to use.



You can't buy Simplex™, but it is now available as an integral part of
it's my **Business**™ and will be used by it's my **Word**™, it's my **Graphics**™, . . .

Businessmen! *it's my **Business*** will revolutionize the way that you handle your business. It saves time, money, and standardizes your system for all who use it. *it's my **Business*** comes with applications like accounting, interoffice or intraoffice mail, editing, invoicing, inventory management, mail list, calendar, scheduler, forms and more. You can modify each of these to create applications specifically designed for you... maybe we should have called it "it's your Business".

Professionals! *it's my **Business*** has over 200 pages of examples and demonstrations to show you how to solve your everyday professional problems. And if these examples aren't enough, we give you a complimentary one-year subscription to Questalk™, our hands-on Simplex applications magazine.

System integrators and consultants, beware! If you are not using *it's my **Business*** with Simplex to solve your problems, don't be surprised when more novice programmers solve that complex math, industrial engineering, or business problem faster. We think that you can cut your concept-to-development time by an order of magnitude!

*it's my **Business*** (includes *it's my **Editor***) - \$695.00
*it's my **Business*** Demo Disk - \$20.00
*it's my **Editor*** - \$100.00.

Quest Research software is available through your local computer store or through mail order from Quest Software Corporation at (205) 539-8086, 303 Williams Avenue, Huntsville, AL 35801.

Value added resellers and dealers please contact Quest Research, Incorporated at (800) 558-8088, 303 Williams Avenue, Huntsville, AL, 35801.

Quest™
Quest Research Inc.

IBM is a registered trademark of International Business Machines. Macintosh is a trademark of Apple Corporation. *it's my **Business***, *it's my **Word***, *it's my **Graphics***, *it's my **Editor***, *it's my **Home***, *it's my **Voice***, *it's my **Ear***, *it's my **Statistics***, Simplex, MouSim, Questalk, and the Quest logo are trademarks of Quest Research, Incorporated.

Circle no. 57 on reader service card.

NEW from BORLAND! TURBO TOOLBOX & TURBO TUTOR

"TURBO is much better than the Pascal IBM sells."
Jerry Pournelle, Byte, July 1984
"If you have the slightest interest in Pascal—buy it."
Bruce Webster, Softalk IBM, March 1984

BORLAND INTERNATIONAL GIFT PACK

ONLY
\$99.95
A SAVINGS OF \$30!

What a gift for you and your friends! The extraordinary TURBO PASCAL compiler, together with the exciting new TURBO TOOLBOX and new TURBO TUTOR. All 3 manuals with disks for \$99.95.

TURBO PASCAL Version 2.0 (reg. \$49.95). The now classic program development environment still includes the FREE MICROCALC SPREAD SHEET. Commented source code on disk

- Optional 8087 support available for a small additional charge

NEW! TURBO TOOLBOX (reg. \$49.95). A set of three fundamental utilities that work in conjunction with TURBO PASCAL.

Includes:

- TURBO-ISAM FILES USING B+ TREES. Commented source code on disk
- QUICKSORT ON DISK. Commented source code on disk
- GINST (General Installation Program)

Provides those programs written in TURBO PASCAL with a terminal installation module just like TURBO'S!

- NOW INCLUDES FREE SAMPLE DATABASE

NEW! TURBO TUTOR (reg. \$29.95). Teaches step by step how to use the TURBO PASCAL development environment—an ideal introduction for basic programmers. Commented source code for all program examples on disk.

30 DAY MONEY BACK GUARANTEE

These offers good through Feb. 1, 1985

For VISA and MASTERCARD order call toll free:

1-(800)-255-8008 1-(800)-742-1133

(Lines open 24 hrs., 7 days a week)

Dealer and Distributor inquiries welcome (408) 438-8400

CHOOSE ONE (please add \$5.00 for handling and shipping U.S. orders)

_____ All Three-Gift Pack	\$ 99.95 + 5.00 SPECIAL!
_____ All Three & 8087	139.95 + 5.00 SPECIAL!
_____ Turbo Pascal 2.0	49.95 + 5.00
_____ Turbo Toolbox	49.95 + 5.00
_____ Turbo Tutor	29.95 + 5.00
_____ Turbo 8087	89.95 + 5.00

Check _____ Money Order _____ VISA _____ MasterCard _____

Card #: _____ Exp. date: _____ Shipped UPS

My system is: 8 bit _____ 16 bit _____

Operating System: CP/M 80 _____ CP/M 86 _____ MS DOS _____ PC DOS _____

Computer: _____ Disk Format: _____

Please be sure model number & format are correct.

NAME: _____

ADDRESS: _____

CITY/STATE/ZIP: _____

TELEPHONE: _____

California residents add 6% sales tax. Outside U.S.A. add \$15.00 (if outside of U.S.A. payment must be by bank draft payable in the U.S. and in U.S. dollars). Sorry no C.O.D. or Purchase Orders.

G11



4113 Scotts Valley Drive
Scotts Valley, CA 95066
TELEX: 172373

